Optimal Reliability Design of Multilevel Systems Using Hierarchical Genetic Algorithms(Dissertation_全文)

AUTHOR(S):

Kumar, Ranjan

Optimal Reliability Design of Multilevel Systems Using Hierarchical Genetic Algorithms

Ranjan Kumar

Optimum System Design Engineering Laboratory
Department of Aeronautics and Astronautics
Kyoto University
JAPAN

Abstract

The demand for higher reliability tends to make system design increasingly complex. The configurations of such complex design are hierarchical containing multiple layers of subsystems immediately below the system level and component at the lowest levels. Reliability optimization of hierarchical systems can be achieved by allocating appropriate redundancy to unreliable units at all levels; system, subsystems, and component. However, allocating redundancy is subject to economic and physical limitations. The problems of optimal reliability design using redundancy allocation falls into the category of nonlinear integer programming problems which are quite difficult to solve because they are NP-hard and involves discrete design variables.

A comprehensive examination of literature reveals that multilevel redundancy allocation optimization problems are seldom addressed in terms of the detailed modeling or appropriate optimization technique that such problems acquire. Additionally, currently, attention paid, however, to redundancy allocation is mainly confined to a single level, principally due to the notion that redundancy at the component level is more effective than at system level. However, this is not true for redundancy scenarios having non-identical spare parts and large-scale structures. Therefore, to increase the efficiency, reliability and maintainability of sophisticated products, the design engineer has to make a transition from a traditional focus on single level redundancy, and deal more effectively with multilevel redundancy.

This research is a paradigm shift in approach to bridge the gap existing between the current techniques and required techniques of optimal reliability design of multilevel systems. First, this work introduced the scope of hierarchical and modular concepts in optimal reliability design and proposed a general formulation for Multilevel Redundancy Allocation Optimization Problems (MRAOPs) for reliability optimization of hierarchical

system.

Second, this research proposed a new hierarchical genetic algorithm for solving multilevel redundancy allocation optimization problems more efficiently. Because the design variables of MRAOPs are hierarchically structured, this work created a novel hierarchical genotype coding scheme with two types of genes; nodal and terminal. With this new hierarchical genotype coding scheme, hierarchical genetic algorithm is applied to solve several multilevel redundancy allocation optimization problem and found superior in performance to that using conventional genetic algorithms.

Third, this work introduced the concept of modularity in optimal reliability design and solved multilevel redundancy allocation optimization problems with series and series-parallel configuration. The results showed that modular redundancy allocation not only provide better reliability but also make a system more fault tolerant.

Finally, this work proposed a multiobjective formulation and optimization of multilevel redundancy allocation optimization problems. A general framework of Multiobjective Hierarchical Genetic Algorithm (MOHGA) has been proposed to solve multiobjective optimization of MRAOPs. The Non-dominated Sorting Genetic Algorithm (NSGA-II) and the Strength Pareto Evolutionary Algorithm (SPEA2) have been implemented in selection operator of the proposed MOHGA. The result demonstrated that selection operator with non-dominated sorting genetic algorithm performed better than the other methods in solving multiobjective optimization of MRAOPs.

Building on the research carried out here, the future research should focus on designing efficient optimization techniques and on creating better selection strategies for multiobjective genetic algorithms when solving multilevel problems. In sum, this work has opened up a new avenue of future research in the area of optimal reliability design of multilevel systems.

Contents

Figures

Tables

Chapter 1

Introduction

With the heightened quality consciousness faced by industry, modern engineering is now more intended on developing tools and techniques to enhance product and process reliability in the design stage[1]. To remain competitive, the guarantee of high system reliability at a competitive cost is essential. Recent shifts in the focus of sophisticated space and defense operations have opened up many new frontiers of technological challenges and created demands for more reliable and maintainable products. Additionally, unprecedented developments in nanotechnology and miniaturization have led to many complex reliability structures, and increased the complexity of product designs. These developments give rise to a problem of how high reliability can be built into the complex design within the limit of economical and physical constraints. Thus a design optimization discipline which deals with the problems of maximizing reliability in a product design within the available resources is termed optimal reliability design. Engineers largely accomplish the optimal reliability design through the use of better computational optimization models, continuously growing in both complexity and fidelity. In general, achieving optimal reliability design is quite difficult because reliability design optimization problems are NP-hard[2] and involves discrete design variables.

1.1 Optimal reliability design

The availability of powerful computational tools and techniques has shortened the development time and provided engineers with an opportunity to obtain improved designs. Increasingly the engineers are employing optimization as a design tool for finding optimal designs characterized by lower cost while satisfying performance requirements. A typical optimization example includes maximizing reliability while satisfying design cost

constraints. The basic paradigm in design optimization is to find a set of design variables that optimizes an objective function while satisfying the resource constraints. The process of obtaining optimal reliability in a product design is known as optimal reliability design.

Optimal reliability design can be achieved in several ways such as by enhancing component reliability, providing redundant components in parallel, by ensuring the combination of both previous options, and reassignment of interchangeable components[1]. However, design-reliability experts have focused a great deal of their efforts on allocation of reliability and redundancy of components for maximizing system reliability. Development of efficient solution methods for maximizing system reliability by allocation still remains a challenging task in design for reliability.

1.1.1 Overview of reliability design optimization

The research interest in quantitative aspects of optimal reliability design began in the 1960s. Since then many credible approaches for optimal reliability design have been proposed. The review papers[3][4][5][6] of the earlier work indicate that the diversity of system structures, resource constraints, and options has led to the creation of several optimization models. The research works in the area of reliability design optimization can be classified[6] using three criteria: system configuration, problem type, and optimization methodologies. Out of these three criteria of research classification, system configuration and optimization methodologies are still evolving areas of research in optimal reliability design of sophisticated systems.

System configuration, also called reliability block diagrams (RBD), depicts the logical relationship between the functioning of the system and the functioning of its components. RBD of a system actually represents the real world system structures and quite often influences the optimization approach used in optimal reliability design. All the research works in this area can be grouped into these major system structures: series[7-15],

parallel[11,12,14,16-21], parallel-series[22-63], general network[64-77], *k*-out-of-*n*: G(F) systems[78-87], and other unspecified configurations [88-93]. However, reliability optimization of hierarchical RBDs has hardly been a research focus in optimal reliability design. It is well known that almost all of the complex engineering systems are hierarchically configured in which the system level is at the top, subsystem levels in the middle, and component level at the lowest level. The optimization problems of hierarchical RBD is a class of multilevel allocation optimization problems and very hard to solve these problems particularly using available optimization techniques.

Based on the review works by Tillman et al.[4], Misra[5], and Kuo and Prasad[6], the major focus of the research in optimization methods for system-reliability optimization can be classified into different categories such as heuristic algorithms, metaheuristic algorithms, exact methods, heuristics for reliability-redundancy allocation, and multiple objective reliability optimization. Almost all of the heuristics developed before 1980, obtained solutions from the solution of a previous iteration by increasing one of the variables by 1. Selection of the variable for the increment is based on a sensitivity factor. Nakagawa and Miyazaki[46] numerically compared the heuristic methods of Nakagawa and Nakashima[47], Kuo et al.[71], Gopal et al.[34], and Sharma and Venkateswaran[56] for a redundancy allocation problem with nonlinear constraints.

However, the heuristic developed after 1980 are based on distinct approaches. Dinghua Shi[94] developed a heuristic method for optimum redundancy allocation with separable, monotonic nondecreasing constraint functions following the approach of adjusting unit increment with time. For the similar problem, Kohda and Inoue[70] developed a heuristic method in which the solutions of two successive iterations may differ on one or two variables. Kim and Yum[68] developed a heuristic algorithm with separable, monotonic nondecreasing constraint functions. Kuo at al.[41] presented a heuristic method based on a

branch-and-bound strategy and Langrange multiplier method. Recently, Jianping[38] recently developed the bounded heuristic method for optimal redundancy allocation in which the method moves from one bound point to another through an increase of 1 in a selected variable and changes in some variables.

Because exact methods provide exact optimal solution to a problem and involve more computational efforts, researchers in general have directed very little attention toward exact solution methodologies for such problems. Such methods are particularly advantageous when the problem is not large. Tillman et al. [3] documented many exact methods developed before 1980. Nakagawa and Miyazaki[47] adopted the surrogate constraints method to solve optimum redundancy allocation optimization problem when there are two constraints, and objective as well as the constraint functions are separable. Misra[95] adopted proposed an exact algorithm for optimal redundancy allocation based on a search near the boundary of the feasible region. This method was later implemented by Misra and Sharma[43], Sharma et al.[77], and Misra and Misra[96] for solving various problems. Prasad and Kuo[93] recently developed a partial enumeration method based on a lexigraphic search with an upper bound on system reliability. For large systems with a good modular structure, Li and Haimes[73] proposed a three-level decomposition method for reliability optimization subject to resource constraints. Mohan and Shankar[74] adopted a random search technique for finding a global optimal solution to the problem of maximizing system reliability through the selection of only component reliabilities subject to cost constraints. Bai et al.[78] considered a *k*-out-of-*n*: G system with common cause failures.

Providing redundancy and enhancing component reliability at the same time often lead to increase in system cost. Such problems are termed reliability-redundancy allocation problem and belong to a class of nonlinear mixed integer programming problems. Tillman et al.[59] were among the first to solve the problem using a heuristic and search technique.

Gopal et al.[35] developed a heuristic method that starts with 0.5 as the component reliability at each stage of the system, and increases component reliability at one of the stages by a specified value h in every iteration. Xu et al.[61] offered an iterative heuristic method for such problems with separable constraints. Hikita et al.[67] developed a surrogate constraints method to solve reliability-redundancy allocation optimization problems with separable constraints. Reliability-redundancy allocation optimization problems arise in software also. Chi and Kuo[90] formulated mixed integer programming problems for such allocation in software systems and systems involving software and hardware.

Multiobjective optimization problems are adopted when there are several conflicting objectives are present and optimizing all these objectives simultaneously. The approach usually involves determination of all Pareto optimal solutions. Sakawa[55] adopts a large-scale multiple objective optimization method to deal with the problem of determining optimal levels of component reliabilities and redundancies. In this approach, he derives Pareto optimal solutions by optimizing composite objective functions, which are obtained as linear combinations of the four objective functions. Later, Sakawa[97] provides a theoretical framework for the sequential proxy optimization technique (SPOT); which is an interactive, multiple objective decision-making technique for selection among a set of Pareto optimal solutions. Misra and Sharma[44] adopt an approach which involves the Misra integer programming algorithm and a multi-criteria optimization method based on the min-max concept for obtaining Pareto optimal solutions. Misra and Sharma[45] also presented a similar approach to solve multiple objective reliability-redundancy allocation optimization problems. Dhingra[30] adopts another multiple objective approach to maximize system reliability and minimize consumption of resources: cost, weight, and volume.

Most of the system-reliability optimization problems fall into the category of nonlinear integer programming problems. Because the solutions of these problems must be integers,

they are more difficult to solve than general nonlinear programming problems. Though there are several optimization techniques have been applied in solving reliability optimization problems, not a single method is able to solve all reliability optimization problems. For example, dynamic programming[98-101] has dimensionality difficulties which increase with increasing number of state variables, and it is hard to solve problems with more than three constraints. Although integer programming[99, 102-107] methods yields integer solutions, transforming nonlinear objective functions and constraints into linear forms is a difficult task and they do not guarantee that optimal solutions can be obtained in a reasonable time. Exact algorithms[108-114] such as branch-and-bound and other implicit enumeration methods require much computational effort to determine an exact solution.

Although many algorithms have been proposed for nonlinear programming problems, only a few, such as sequential unconstrained minimization technique(SUMT)[115-117], the modified sequential simplex pattern search[16], and the generalized Langrangian function method[118-120], have been proved to be effective when applied to large-scale reliability optimization problems. The maximum principle has difficulty in solving problems with more than three constraints. Likewise geometric programming is restricted to problems that can be formulated by polynomial functions. Unlike all these optimization techniques, meta-heuristic approaches, such as GA[121, 122], simulated annealing methods[123], tabu search methods[124], particle swarm optimization method[128] have been found to be very flexible and versatile in solving reliability optimization problems. They are based more on artificial reasoning than classical mathematics-based optimization. They require fewer assumptions on the objective as well as the constraint functions.

In reliability design optimization, metaheuristic algorithms have been successfully applied to solve varieties of problems. A good description of the GA concept and its application in reliability optimization is described in later chapter. The simulated annealing

algorithm is a general method used to solve combinatorial optimization problems. It involves probabilistic transitions among the solutions to the problem. Cardoso et al.[126] introduced the non-equilibrium simulated annealing algorithm (NESA) by modifying the algorithms of Metropolis et al.[127]. Ravi et al.[76] have recently improved NESA by incorporating a simplex-like heuristic in the method and applied it to solve reliability optimization methods. Another metaheuristic algorithm Tabu search is very useful for solving large-scale complex optimization problems. The salient feature of this method is the utilization of memory to guide the search beyond local optimality. However, Tabu search is the difficulty involved in defining effective memory structures and memory-based strategies which are problem-dependent. Despite a few drawbacks, GA is more popular among rest of the metaheuristic algorithms and this research develops new GA to solve a proposed hierarchical optimization problems.

1.1.2 Reliability Design Optimization using GA

Optimization of system reliability is in general a highly complex problem in which the objective functions as well as the constraints are nonlinear and the decision variables are integers. Such problems are difficult to solve and computationally time consuming. The major research issue in this area is to develop simple heuristics which can give near-optimal solutions with less computational effort. However, heuristic methods usually require a mathematical formulation of the problem and do not provide much tradeoff between quality of solution and computational effort. In contrast to heuristic methods, a GA can be designed for a problem without explicit mathematical formulation, and the values of its parameters can be appropriately chosen to balance both quality of the solution and the computational work.

A conventional genetic algorithm solves a problem by imitating the natural evolution process in which populations undergo continuous upgrade through four process, namely

evaluation, selection, crossover, and mutation. Fist, the design variables are encoded by genotypes and evaluated during evaluation process to calculate fitness. The selection process deletes the individuals with low fitness and retains the individuals having high fitness. Then, selected individuals go through the crossover operation in which genes are exchanged between these individuals and thus the new individuals are generated. Finally, the mutation operator randomly changes the values of genes and generates new individuals. This sequence is iterated till a stopping criterion is met. Keeping the imitation of natural evolution as the foundation, genetic algorithms can be appropriately designed and modified to exploit special features of the problem to be solved.

Recently, several genetic algorithms have been developed and applied to solve a variety of reliability optimization problems[122-127]. A good description of genetic algorithms used for solving reliability optimization problems can be found in literature[1,6,128]. Moreover, a brief survey on GA-based approach[129] for various reliability optimization indicates increased focus on designing hybrid GA[130-134] by combining GA with neural network, fuzzy logic, and other conventional search technique. The successful application of GA-based approaches in solving reliability optimization problems demonstrated the following advantages offered by them. First, genetic algorithms are non-gradient methods, which rely on objective function values and do not require sensitivity analysis. Second, genetic algorithms show high performance in solving multi-peak problems. A group of individuals can be used in a single optimization process, where crossover and mutation operators work to sustain a variety of individuals distributed across the searching space, and convergence to false local optima is avoided. Third, genetic algorithms work with a coding of solution set. This feature of GA is a powerful which let one to develop innovative genotype representation of design variables. This work proposes a new encoding method by exploiting this feature of genetic algorithms.

Almost all of the GA-based approaches were applied to solve reliability optimization problems confined to single or double levels of series, parallel, parallel-series, general network, *k*-out-of-*n*:G(F), and the other configurations. However, the reliability optimization of hierarchical system with more than two levels has hardly been extensively dealt with in optimal reliability design. Reliability optimization of hierarchical structures falls into the class of hierarchical optimization problems having hierarchical design variables and the optimization problems are termed as multilevel redundancy allocation optimization problems. Recently, the growing research interest in multilevel reliability modeling and optimization using GA, which will be discussed in later chapters of this work, is reflected in the literature, due to the practical importance of these techniques. However, almost all of these GA-based approaches applied artificial transformation of multilevel design variables into vector representations, because conventional GA uses one-dimensional representations of design variables. Unfortunately, the additional constraints imposed when transforming hierarchical design variables into vector design variables artificially constrict the feasible design region, often leading to suboptimal solutions.

1.2 Research objectives

This dissertation investigates and develops formulations and methodologies for multilevel redundancy allocation optimization problems (MRAOPs) used in optimal reliability design of complex systems. The main focus is to propose a generic formulation of MRAOP and develop methodologies that yield near global solutions and superior to those solution obtained using conventional genetic algorithms. Efforts are focused on developing new genetic algorithms for better optimal solutions by proposing innovative hierarchical genotype representation scheme for hierarchical design variables of MRAOPs. A generalized multilevel formulation in which redundancy can be allocated to any units at

all levels without imposing any artificial restrictions, is developed. In addition to the multilevel formulation and hierarchical genetic algorithms, a novel framework of multiobjective hierarchical genetic algorithms for multiobjective optimization of MRAOPs is developed to address the concerns of artificial transformation of hierarchical design variables into vector and existing genetic operator's inability to handle elitism with hierarchical genotype codes. In these investigations two popular multiobjective genetic algorithms (MOGAs) - the strength Pareto evolutionary genetic algorithm (SPEA2) and the nondominated sorting genetic algorithm (NSGA-II) have been implemented in the selection operators of the newly developed general framework of multiobjective hierarchical genetic algorithms.

1.2.1 Hierarchy and modularity in optimal reliability design

This work presents hierarchical and modular concepts and their application in optimal reliability design, and proposed a hierarchical formulation of reliability design structures. In this formulation, there are multiple levels of hierarchy in system design and redundancy can be allocated to any unit and at any level when maximizing system reliability. Modular concept for better design is well established and modular reliability designs not only enhance the system reliability but also make more fault tolerant. Therefore, to increase the efficiency, reliability and maintainability of a multilevel reliability system, the design engineer has to shift away from the conventional focus on component redundancy, and deal more effectively with issues pertaining to modular redundancy. This work proposed a formulation of modular optimization scheme for multilevel redundancy allocation optimization problems. This chapter proposed a generalized formulation for multilevel redundancy allocation problems that can handle redundancies for each unit in a hierarchical reliability system, with structure containing multiple layers of subsystems and components. Multilevel redundancy allocation is an especially powerful approach for

improving the system reliability of such hierarchical configurations, and system optimization problems that take advantage of this approach are termed multilevel redundancy allocation optimization problems (MRAOP).

1.2.2 Hierarchical genetic algorithms for multilevel redundancy allocation optimization

This research designed and developed a hierarchical genetic algorithm (HGA) that uses special genetic operators to handle the hierarchical genotype representation of hierarchical design variables. Because the design variables in MRAOP are hierarchically structured, this work proposes a new variable coding method in which these hierarchical design variables are represented by two types of hierarchical genotype, termed ordinal node, and terminal node. These genotypes preserve the logical linkage among the hierarchical variables, and allow every possible combination of redundancy during the optimization process. For comparison, the customized HGA, and a conventional genetic algorithm (GA) in which design variables are coded in vector forms, are applied to solve MRAOP for series systems having two different configurations. The solutions obtained when using HGA are shown to be superior to the conventional GA solutions, indicating that the HGA here is especially suitable for solving MRAOP for series systems.

1.2.3 Optimal modular redundancy allocation in series and series-parallel systems

To achieve truly optimal system reliability, the design of a complex system must address multilevel reliability configuration concerns at the earliest possible design stage, to ensure that appropriate degrees of reliability are allocated to every unit at all levels. However, the current practice of allocating reliability at a single level leads to inferior optimal solutions, particularly in the class of multilevel redundancy allocation problems. Multilevel redundancy allocation optimization problems frequently occur when optimizing the system reliability of multilevel systems. It has been found that a modular scheme of

redundancy allocation in multilevel systems not only enhances system reliability but also provides fault tolerance for the optimum design. This research proposes a method for optimizing modular redundancy allocation in two types of multilevel reliability configurations, series and series-parallel. A modular design variable is defined to handle modular redundancy in these two types of multilevel redundancy allocation problem. A customized genetic algorithm, namely, a Hierarchical Genetic Algorithm (HGA), is applied to solve the modular redundancy allocation optimization problems, in which the design variables are coded as hierarchical genotypes. The numerical examples solved in this chapter demonstrate the efficacy of a customized HGA for multilevel system reliability optimization. Additionally, the results obtained in this chapter indicate that achieving modular redundancy in series and series-parallel systems provides significant advantages when compared with component redundancy. The demonstrated methodology also indicates that future research may yield significantly better solutions to the technological challenges of designing more fault-tolerant systems that provide improved reliability and lower lifecycle cost.

1.2.4 Multiobjective hierarchical genetic algorithms for optimal reliability design

This research work proposes a multiobjective formulation of MRAOPs and a methodology for solving such problems. In this methodology, a hierarchical GA framework for multiobjective optimization is proposed by introducing hierarchical genotype encoding for design variables. In addition, we implement the proposed approach by integrating the hierarchical genotype encoding scheme with two popular multiobjective genetic algorithms (MOGAs) - the strength pareto evolutionary genetic algorithm (SPEA2) and the nondominated sorting genetic algorithm (NSGA-II). In the provided numerical examples, the proposed multiobjective hierarchical approach is applied to solve two hierarchical MRAOPs, a 4-level problem and a 3-level problem. The proposed method is compared

with a single objective optimization method that uses a hierarchical genetic algorithm, also applied to solve the 3- and 4-level problems. The results show that a multiobjective hierarchical GA that includes elitism and mechanism for diversity preserving performed better than a single objective GA that only uses elitism, when solving large-scale MRAOPs. Additionally, the experimental results show that the proposed method with NSGA-II outperformed the proposed method with SPEA2 in finding useful Pareto optimal solution sets.

1.3 Overview of the dissertation

This dissertation is organized as follows. In chapter 2, the concepts of hierarchy and modularity in optimal reliability design are described. Mathematical formulations of multilevel reliability configurations are proposed. Modular scheme of redundancy allocation optimization is proposed and discussed. Numerical examples and their results are discussed.

Chapter 3 presents an overview of the multilevel redundancy allocation optimization using genetic algorithms. The details of MRAOPs for series system with cost function are described. A new hierarchical genetic algorithm is proposed with innovative encoding methods for hierarchical genetic algorithms. Numerical examples and their solutions are summarized and discussed.

Chapter 4 presents a multilevel reliability design optimization formulation in series and series-parallel systems. Some background works in this area by other researchers along with some issues related to reliability design optimization using genetic algorithms is detailed.

In chapter 5, a new multiobjective hierarchical genetic algorithm is developed. The multiobjective optimization problems of multilevel redundancy allocation are formulated and solved using the proposed algorithms. While solving MRAOPs, this research

implemented two popular multiobjective genetic algorithms (MOGAs) - the strength pareto evolutionary genetic algorithm (SPEA2) and the nondominated sorting genetic algorithm (NSGA-II). The numerical results demonstrated the improvement in optimal solutions using the proposed algorithms.

In chapter 6, the advantages and limitations of the formulation of multilevel reliability design optimization and GA-based methodologies developed in this investigation are presented. Important conclusions are drawn and some future work in this area is recommended.

Chapter 2

Hierarchy and modularity in optimal reliability design

2.1 Introduction

System configuration, also termed reliability block diagram (RBD), is an important attribute that affects optimal reliability designs considerably. To meet the demand of highly reliable systems, designs are becoming increasingly complex and the number of components has increased manifold. Unprecedented development in miniaturization technology, also, has led to more complex designs with new configurations altogether. All these evolving trends in technological development created the necessity of developing more effective and generic configurations to represent complex reliability designs and optimizing such effective configurations. One of such effective structures is hierarchical structure that can not only address the issue of scalability in large scale designs but also offer precision in representing all the components in a complex design. Though hierarchical configuration of reliability structure is already in practice, the theory of hierarchical RBD is yet to be dealt with in optimal reliability design. This work presents the hierarchical and modular concepts and its importance in optimal reliability design, and proposes a general formulation of hierarchical RBD for multilevel redundancy allocation.

2.2 Hierarchical systems

Hierarchical systems are composed of subsystems each of which is a hierarchical by itself until the bottom level[135]. Hierarchical systems contain multiple levels and at each level, the interactions within each subsystem are of much higher magnitude than the interactions between the subsystems. This property of hierarchical systems is called near decomposability[135] that can help to design complex system easily. Hierarchical systems are omnipresent around us. An excellent example includes in biological systems, organisms

are composed of organs, organs are composed of tissues, tissues are composed of cells, and so on.

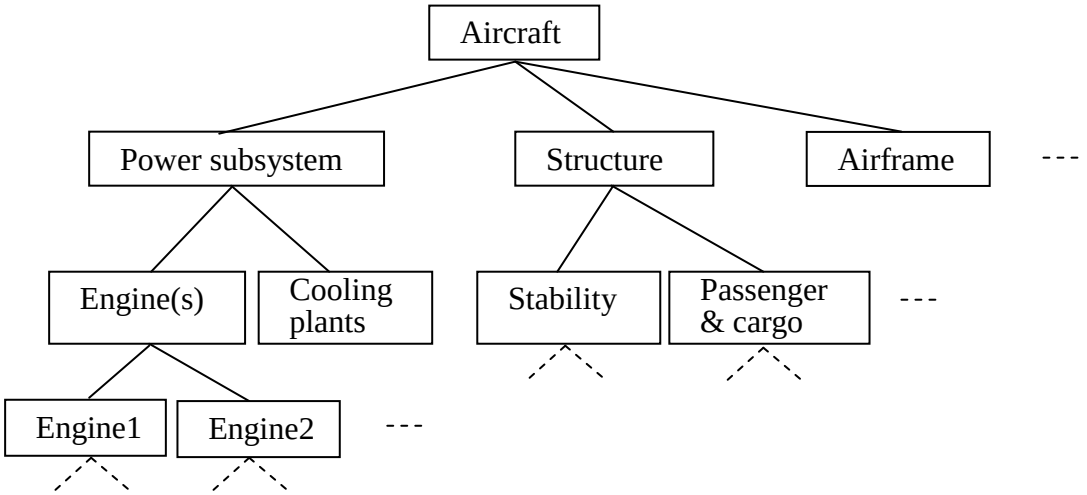


Fig.2.1 Hierarchical design of an aircraft.

The concept of hierarchical system helped also helped in solving complex problems of science and engineering[136]. In software engineering, for instance, complex software systems are built on multiple levels. At each level, components (functions, libraries, objects, etc.) from lower levels are used as basic building blocks to construct new components, and those components are, in turn, used at even higher levels. Starting from assembly language to the sequential query language of database systems, hierarchical design allow us to develop complex systems that could not be approached at a single level.

This work intends to describe the hierarchical approach of an aircraft design and applies this approach to solving difficult reliability optimization problems of practical systems. As shown in Fig. 2.1, an aircraft is composed of power subsystem, structure, airframe, and so forth. Each of these subsystems can be further decomposed. For example, the power subsystems contain an engine or engines, cooling plants, and so on, and the individual engine can be further decomposed. The design is hierarchical in its nature, and each component of the car can be further decomposed into a number of subsystems. All the subsystems work in combination with the purpose of flying.

A system is composed of hardware and software to perform certain function. In a large system, numerous components make a design more complex. However, a proper decomposition significantly simplifies the design. For example, instead of designing "an aircraft that flies," we design subsystems capable of producing rotational movement (engine), providing lift(airframe), stability (structure), slowing down the movement (braking system), and so on. This simplification can go down a number of levels and makes the task of building an aircraft much easier than if viewed on a single level. This leads to the consideration of hierarchical decomposition of large scale reliability design and simplification of the optimization process by allocating optimum redundancy to subsystems at each hierarchical level.

2.3 Hierarchical reliability block diagram

This section intends to describe the subtleties of hierarchical structure of RBD and shows how hierarchical arrangement enables a design engineer to represent the reliability design of large scale systems in a simpler way and analyze it more accurately. To deal with the issue of simplifying the reliability design of complex system, hierarchical reliability block diagram is an indispensable tool, which is a graphical representation of the system reliability structure composed of system at the top level, subsystems at the middle levels, and components at the lowest levels. As defined in the introduction section, a RBD defines the logical interaction of failures within a system. Individual blocks may represent single component failures, sub-system failures and other events that may contribute towards system failures. The reliability behavior of an individual sub-system block may be represented by a RBD at a lower hierarchical level. The logical flow of a RBD originates from an input node at lower side of the unit to an output node at the upper side of the unit. Blocks are arranged in series and parallel arrangements between the system input and output nodes.

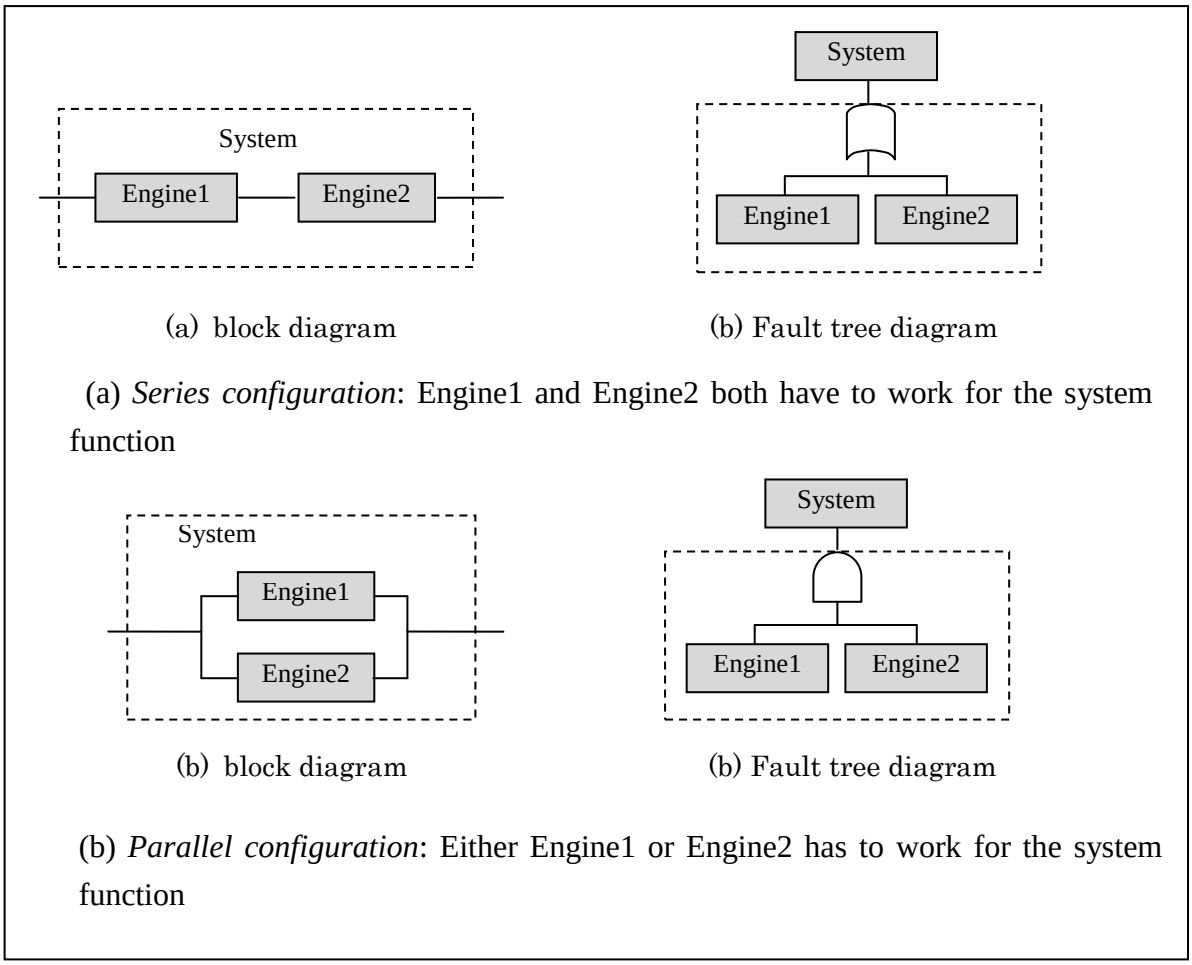


Fig. 2.2. Building blocks of hierarchical RBD.

Fig. 2.2 shows the series and parallel relationship between engine1 and engine2 that forms a system. These two arrangements are also termed building blocks in a hierarchical RBD. A series system works if and only if every component works. Such a system is failed whenever any component is failed. The structure function of a series system is given by

$$R_s = \prod_{i=1}^{n} R_i \tag{2.1}$$

where, R_s, R_i, and n are system reliability, reliability i-th block, total number of blocks in series connection.

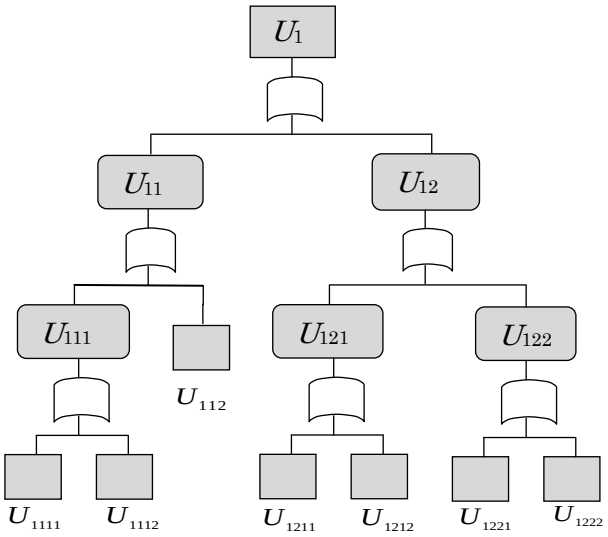
In a parallel system, not all components are necessary for the system to work properly. It works as long as at least one component works. The structure function of a parallel system is given by

$$R_s = 1 - \prod_{i=1}^{n}(1 - R_i) \tag{2.1}$$

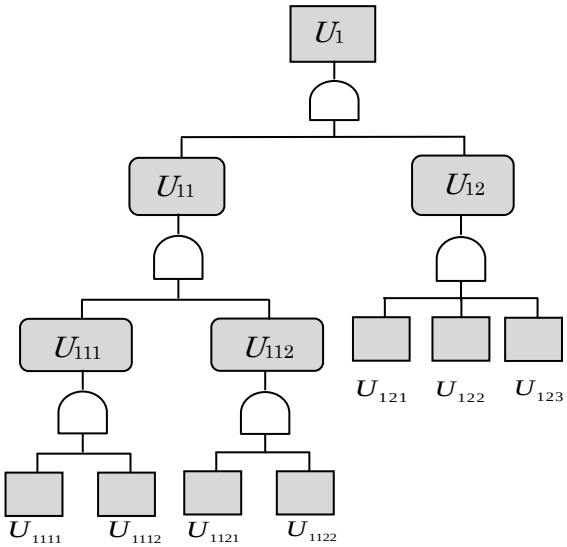
where, R_s, R_i, and n are system reliability, reliability i-th block, total number of blocks in series connection. In a parallel system, only one component needs to work properly to make the system work properly. Therefore, $n-1$ components in the parallel system of n components are called redundant components. They are included to increase the probability that there is at least one working component. Redundancy is a technique widely used to enhance system reliability.

This research proposes bi-level series and bi-level parallel subsystems as building blocks that make up a hierarchical RBD. These series and parallel building blocks are shown in Fig.2.2. With the combination of these two building blocks, almost all hierarchical RBD can be constructed. Fig.2.3 shows the examples of various hierarchical structure of RBD. Each block in the RBD represent system, subsystems, and components and we call them unit. From now onward, a unit denotes either a system or a subsystem or a component. In a hierarchical system, the input of a unit is fed from the output of its immediate lower unit or its children unit. The reliability values and the logical relationships of its child units are used to calculate the reliability of a parent unit. Therefore, the system reliability depends on the reliability values of subunits and components of the system.

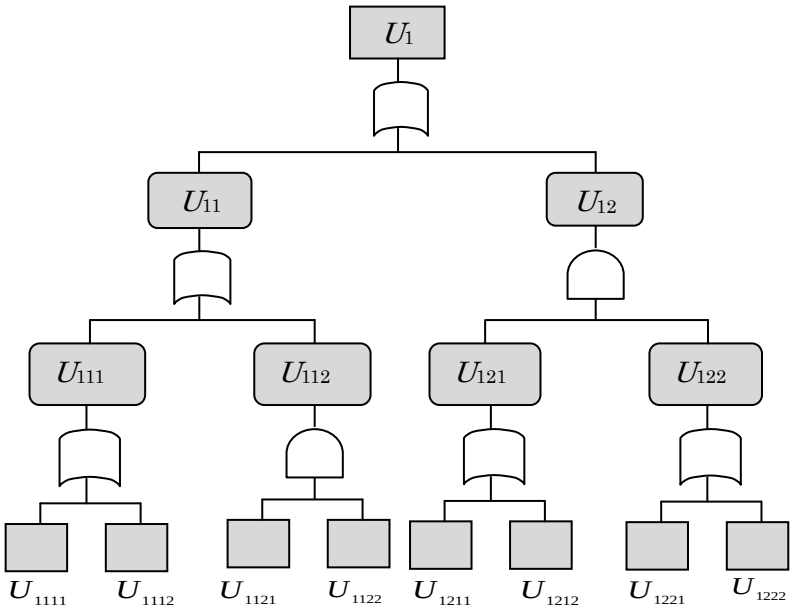
In a hierarchical series system, as shown in Fig.2.3(a), all units must work together at all levels for functioning of the system. In a hierarchical parallel system, all units must work together in a connecting line from lowest level to top level for functioning of the system. In Fig.2.3(b), for instance, U_{11}, U_{111}, and U_{1111} in a connecting line or U_{11}, U_{111}, and U_{1112} in a connecting line have to work together for functioning of system level unit U_1. Fig.2.3(c) shows a complex hierarchical structure of RBD, which is a combination of series and parallel building blocks. Thus, hierarchical structure of RBD presented here can simplify the reliability design of complex system with more accuracy. This is true particularly in optimal reliability design of large scale systems.



(a) Hierarchical series



(b) Hierarchical parallel



(c) Hierarchical series-parallel

Fig. 2.3. Different types of hierarchical structures of RBD.

2.4 Modular design concepts

Modularity is a proved technique for organizing and simplifying a complex system which can contain thousands of different components that function interdependently, while certain components are used only for a specific set of subtasks within the system. Such components of an independent function set can be accommodated within a simple

subsystem, or sub-unit. Here, such a subsystem is called a module. Systems that have modular subsystems usually have superior fault tolerance, ease of maintenance, and are easier to recover at the end of their useful life. Furthermore, a modular system is often simpler than a complex system built from only components.

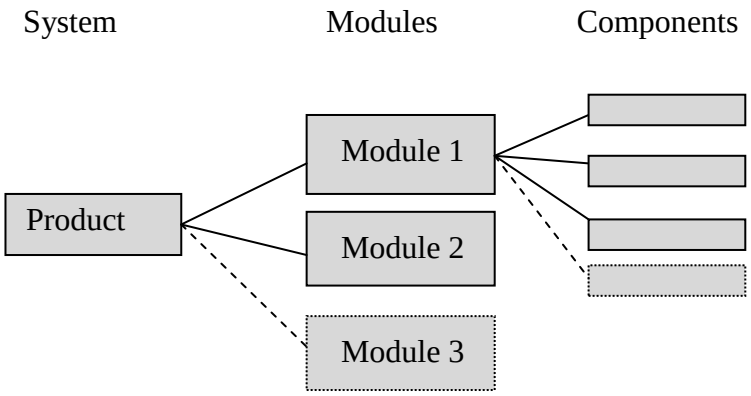
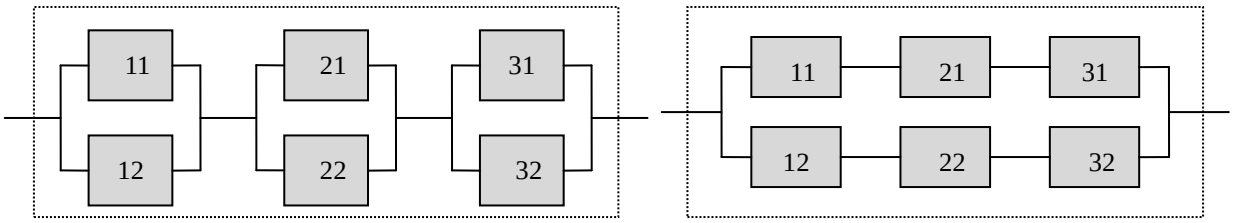


Fig. 2.4. Modules in a hierarchical RBD.

In engineering terminology, a module is a cluster of components that is treated as a single entity in a piece of equipment, as shown in Fig.2.4. In system reliability theory, a module indicates a group of components that has a single input from, and a single output to, the rest of the system. The contribution of all components in a module to the performance of the whole system can be represented by the state of the module. Once the state of the module is known, one does not need to know the states of the components within the module to determine the states of the system. Traditionally, redundancy is added either to a component level or to a subsystem level, when optimizing system reliability. Fig. 2.5 illustrates these two redundancy schemes.

A redundant module is a module added parallel to the existing module to increase its reliability without altering its internal structure. In other words, we preserve a module's internal structure, such as the arrangement of its sub-modules and components, while providing modular redundancy. Thus, to know the status of the system, we need not know the status of its components. Modular redundancy therefore simplifies the complexity of

the system and makes it easier to isolate faults in case of failure.



(a) Redundancy at the component level (b) Redundancy at module level

Fig. 2.5 Redundancy allocation at component and module level.

Under certain assumptions, it is a well-established fact that redundancy at the component level is more effective than redundancy at the system level, but this is not always the case[x5]. Modular or subsystem level redundancy allocation in a large scale system yield superior results particularly for a repairable system. In addition, modular redundancy can help a system become truly fault tolerant. For example, a modular system can shift operation from failed modules to healthy ones, while repairs are carried out. The design transition from component to modular redundancy actually reduces costs and enhances efficiency, flexibility, and reliability.

This is clear from the case study of disk drive presented in a white paper [137] for designing a data storage system more fault tolerant. In 1988, Berkeley researchers presented a landmark paper, “A case for redundant arrays of inexpensive disks(RAID),” proposing several data-writing schemes (“RAID levels”) that such arrays could use to store, retrieve, and recover data. In 1990, the personal computer industry introduced 5.25-inch disks, which had evolved to the point where they had the capacity, performance, and reliability to be used in the first RAID arrays. These new modular storage devices offered a choice of tradeoffs between redundancy and read/write speed, and occupied a fraction of the floor space of the mainframe storage devices they replaced. Thus the modular advantages of RAID arrays are ability to scale up, simpler process of duplication,

specialization of the function of modules, rapid adaptation to the environment, and fault tolerant. Thus modularity in optimal reliability design will have an additional advantage of making the system more tolerant apart from maximizing the system reliability. This is a subtle advantage of modular optimization that this research will achieve.

2.5 General formulation of multilevel redundancy allocation problems

From above description of the concepts of hierarchy and modularity in optimal reliability design, this is clear that large scale and complex system can be dealt with effectively by utilizing these design concepts when solving reliability optimization problems. To maximize the system reliability, there are two ways; one is to enhance the inherent reliability of each component and the other is to provide redundancy to the unit which has poor reliability. The first method has technological limitation and costly beyond a certain point. On the other hand, redundancy allocation is widely practiced in industry for optimal reliability design. Numerous techniques for several structure have been proposed but hierarchical structure is a not thoroughly addressed in terms of mathematical formulation and optimization methodology. This work presents a generalized formulation of a hierarchical redundancy allocation. Since a hierarchical structure of a large scale system contain multiple levels and redundant units are allocated to multiple levels, we term the problems of redundancy allocation in hierarchical systems as multilevel redundancy allocation problems.

The generalized multilevel redundancy allocation formulation proposed here can handle redundancy at every hierarchical levels of a complex system. In this general formulation, a hierarchical structure of RBD of a complex system requires all modules or components at different hierarchical levels to be connected logically either in series or in parallel. For instance, in a basic structure of such a hierarchical series system, all the modules and components at different hierarchical levels are also in series. The basic

structure of a hierarchical RBD means a structure which does not have a redundant unit at any levels. As described before, this basic structure is actually made up of the building block of bi-level hierarchical series and parallel system.

The proposed redundancy model contains multiple hierarchical levels. The system level is the topmost level, and the component level is the lowest. Subsystem or module levels are located between the top, and second lowest levels. Each system, module, and component is here termed a unit. Every unit except components can have any number of subordinate elements, such as modules that make up a system, or components that make up a module. These subordinate elements are called sub-units, whereas the next highest hierarchical unit of a sub-unit is called a parent unit.

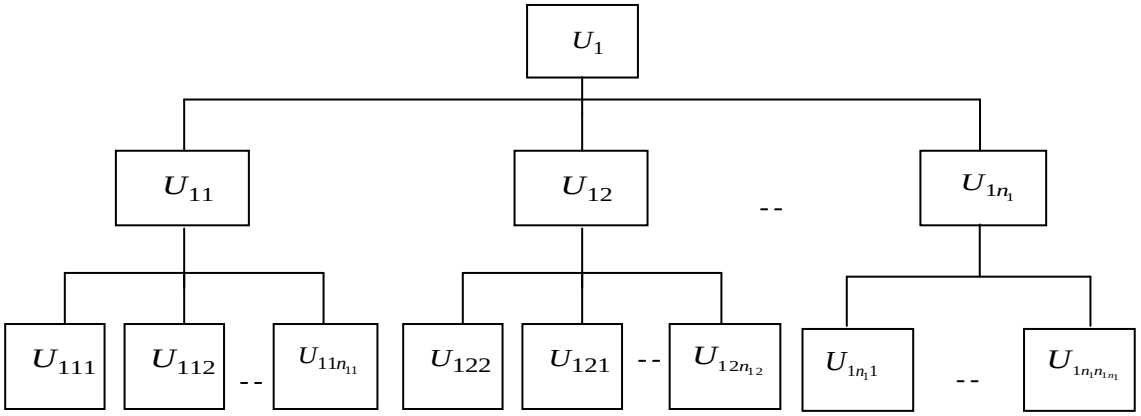


Fig. 2.6. A general multilevel redundancy allocation configuration.

The proposed redundancy allocation model can provide redundancy for all units of a multilevel reliability system. Fig.2.6 represents the schematic diagram of a generalized hierarchical redundancy allocation model. The connecting lines in the diagram imply the logical relationships among the units at different levels, relationships that may be in series, in parallel, or combinations of these two. Redundancy at all levels is assumed to be active, and failures are *s*-independent. Fig.2.7 explains the redundancy allocation scheme in a bi-level series system, and the distinction between sub-units and redundant units. In Fig.2.7(a), U_1 is a unit at the system level that has two sub-units U_{11} & U_{12} at the next

lowest level in the basic configuration. Fig.2.7(b) illustrates the redundancy allocation in U_1, which has two redundant units at system level U_1^1 & U_1^2. Similarly, sub-units U_{11}, and U_{12} have 3, and 1 redundant units, respectively, in parent unit U_1^1, and so on.

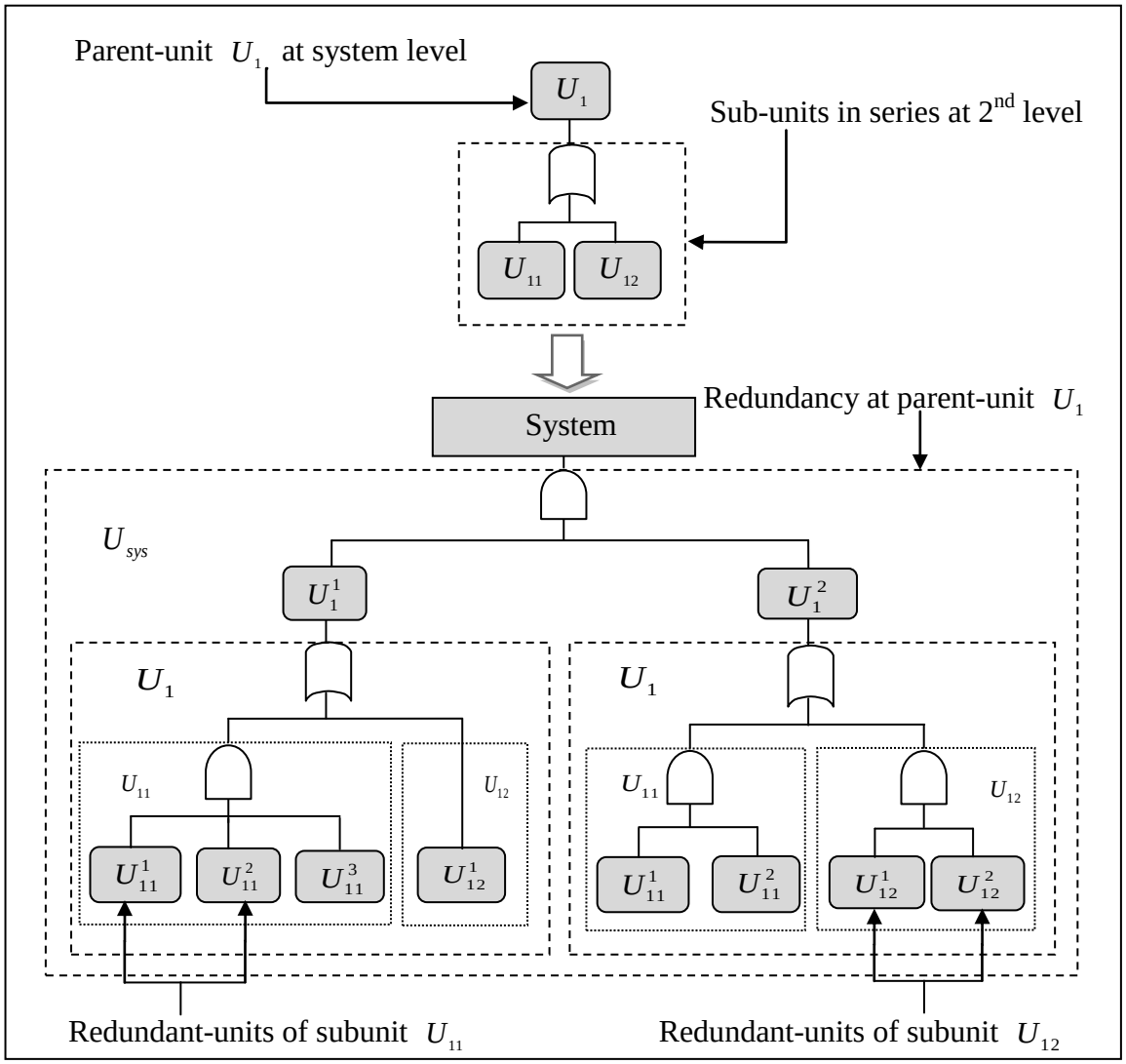


Fig. 2.7. An example of redundancy allocation in bi-level series configuration.

Thus, in a multilevel redundancy allocation model, each unit U_i can have x_i redundant units, and n_i sub-units, so there are $n_i x_i$ sub-units in the level below a parent unit. The sub-units n_i are different for each parent unit in the model described here. For example, as shown in Fig. 2.6, U_1 is a system unit containing U_{11} to U_{1n_1} units as modules at its next lowest hierarchical level. Similarly, the U_{11} module contains n_{11}

sub-units as modules or components at its next lowest level, represented as U_{111} to $U_{11n_{11}}$, which is actually the second level of the system hierarchy. This structure is replicated until the lowest level of the system hierarchy is reached. Thus, the reliability R_i of modular unit U_i for multilevel series configurations can be calculated using

$$R_i = \prod_{m=1}^{n_i}[1 - \prod_{j=1}^{x_i}(1 - R_{i,m}^j)] \tag{2.3}$$

and for multilevel parallel system, the reliability R_i of modular unit U_i can be calculated using

$$R_i = 1 - \prod_{m=1}^{n_i}(1 - \prod_{j=1}^{x_i} R_{i,m}^j) \tag{2.4}$$

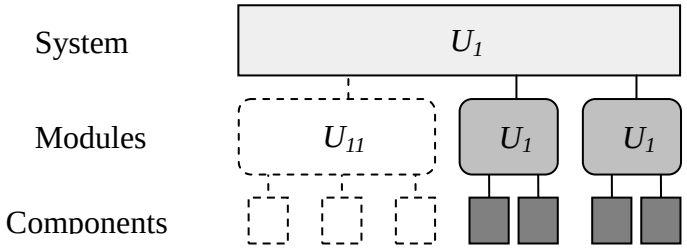
where $R_{i,m}^j$ are reliability values for sub-units $U_{i,m}^j$, a unit in the *j*-th redundant unit of the *m*-th sub-unit of U_i. Each $R_{i,m}^j$ value is calculated using (3) at the level immediately below the unit, and these calculations are recursively iterated to the level just above the very lowest hierarchical level. At the very lowest level, where there are no sub-units belonging to unit U_i, the reliability of component can be obtained as

$$R_i = 1 - \prod_{j=1}^{x_i}(1 - R_i^j) \tag{2.5}$$

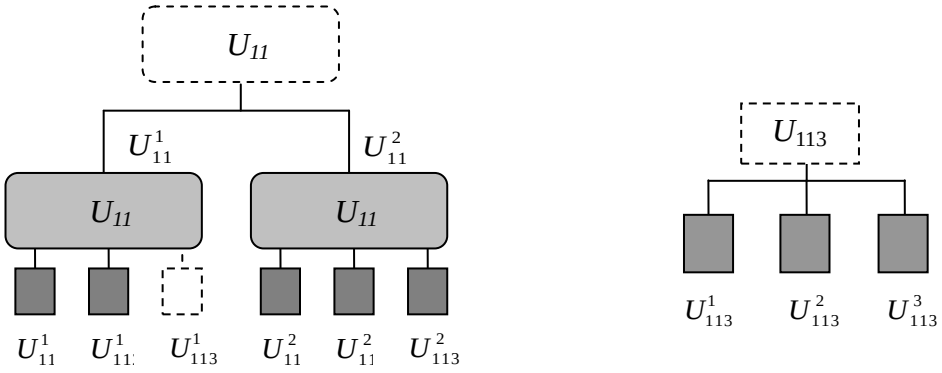
The multilevel reliability allocation model presented here allows redundancy for any unit at any level, and it is thus possible to achieve redundancy schemes that function at both the component, and modular levels. Using (2.3) and (2.5), we can express the mathematical formulation of hierarchical series configuration of hierarchical RBD. Similarly, the combination of (2.4) and (2.5) yield the mathematical expression of hierarchical parallel configuration. To represent hierarchical series-parallel configuration we will need to use all of the three equations.

2.6 Special features of multilevel redundancy allocation formulation

Proposed formulation is a simple way to represent multilevel redundancy allocation scheme in a large scale system. The formulation can exploit the concept of hierarchy and modularity in optimal reliability design of a large hierarchical system. In the proposed redundancy allocation model, redundancy can be allowed to any unit at every level without any constraint. Hierarchical arrangement leads to the simplification of design and provides scalability. Multiple levels between the system and component levels allow modular redundancy which can not only yield superior reliability but also make a system more fault-tolerant.



(a) Basic structure hierarchical RBD of unit U_1



(b) Modular redundancy at U_{11} (c) Component redundancy at U_{113}

Fig. 2.8 Hierarchical series redundancy allocation in a 3-levels system.

For instance, a hierarchical series system shown in Fig. 2.8 is a particular example of the generalized multilevel redundancy allocation model described in Section 2. There are three levels in this HSR system, namely, system, module, and component level. U_1 is a

unit at the system level, (U_{11}, U_{12}, U_{13}) are units at the module level, and (U_{111}, U_{112}, U_{113}, U_{121}, $U_{122}, U_{131}, U_{132}$) are units at the component level. We can calculate the system reliability using Equation (2.3) and (2.5). When a system has redundancy of two or two U_1 modules in parallel, the reliability of U_1 modules is obtained using the following formulation.

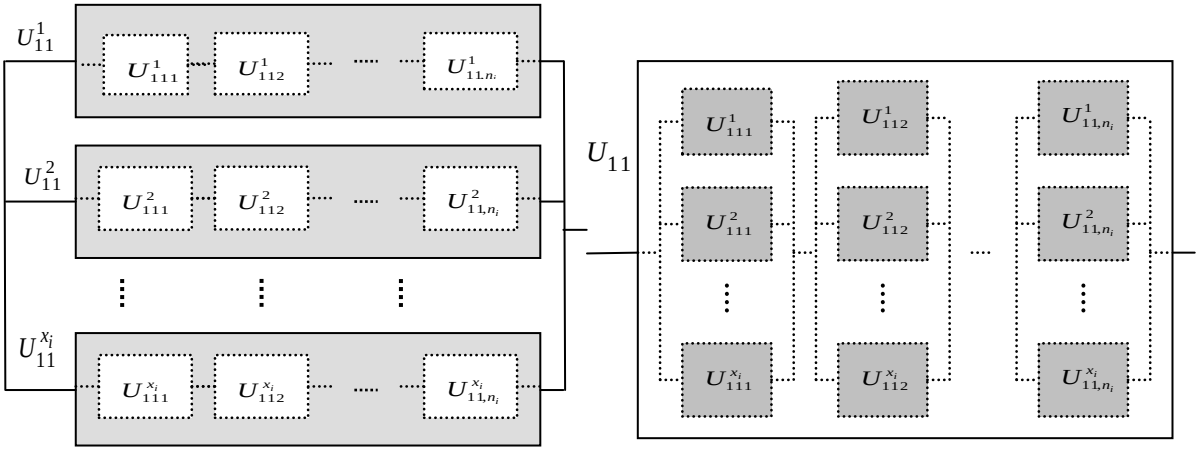
$$R_1 = 1-(1-R^1_{1,1}R^1_{1,2}R^1_{1,3})(1-R^2_{1,1}R^2_{1,2}R^2_{1,3}) \tag{2.6}$$

$R^1_{1,1}$ is the reliability of U^1_{11},which is the first redundant unit of the U_{11}module. When the redundancy of module $U^1_{1,1}$ is three, $R^1_{1,1}$ can be calculated as the following equation

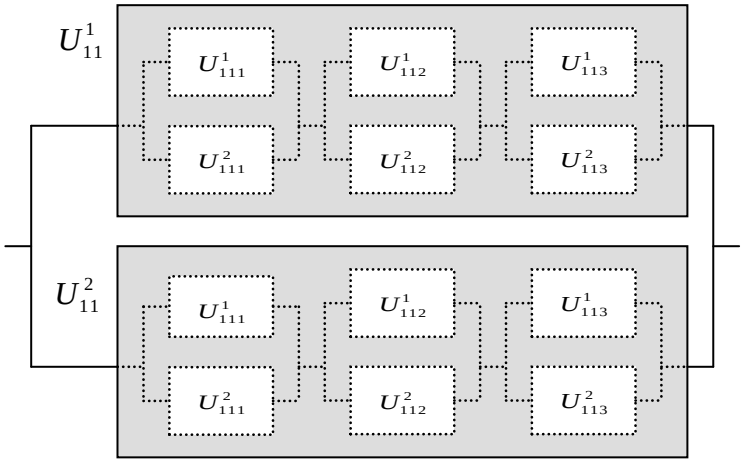
$$R^1_{1,1} = 1-\{(1-R^1_{1,1,1}R^1_{1,1,2}R^1_{1,1,3})\times(1-R^2_{1,1,1}R^2_{1,1,2}R^2_{1,1,3})\times(1-R^3_{1,1,1}R^3_{1,1,2}R^3_{1,1,3})\} \tag{2.7}$$

Similarly, we calculate the reliability of other modules U_{12} and U_{13} in terms of its subunits or components.

As it evident from the Fig.2.8 that proposed multilevel redundancy allocation allows redundancy simultaneously not only at module level but also at component level in the three level hierarchical series system. This is also true for all other hierarchical structures. The proposed hierarchical series systems allow redundancy at any unit, at any level. For instance, Fig. 9 illustrates three types of redundancy in U_{11} modules containing modules or components at its lower levels. Modular redundancy, as shown in Fig.2.9 (a), allows only module-level redundancy during the optimization process. Fig.2.9 (b) shows an example of component redundancy. However, Fig.2.9 (c) illustrates a mixed redundancy in module U_{11}, in which redundancy is possible not only between modules, but also simultaneously among components. Therefore, the mixed redundancy scheme allows the units to have redundancy not only at the same level, but also simultaneously for sub-units at lower levels.



(a) Modular redundancy (b) Component redundancy



(c) Mixed redundancy

Fig. 2.9. Three types of redundancy allocation in a unit U_{11}

The proposed formulation has distinct advantage of representing and maintaining actual configuration that is not available with any other formulation when solving redundancy allocation optimization. Fig.2.10 explains clearly that the hierarchical RBD representation of large scale system is more accurate in optimizing the actual multilevel level redundancy allocation problems than the artificially reduced multilevel configuration into traditional bi-level configuration before optimization.

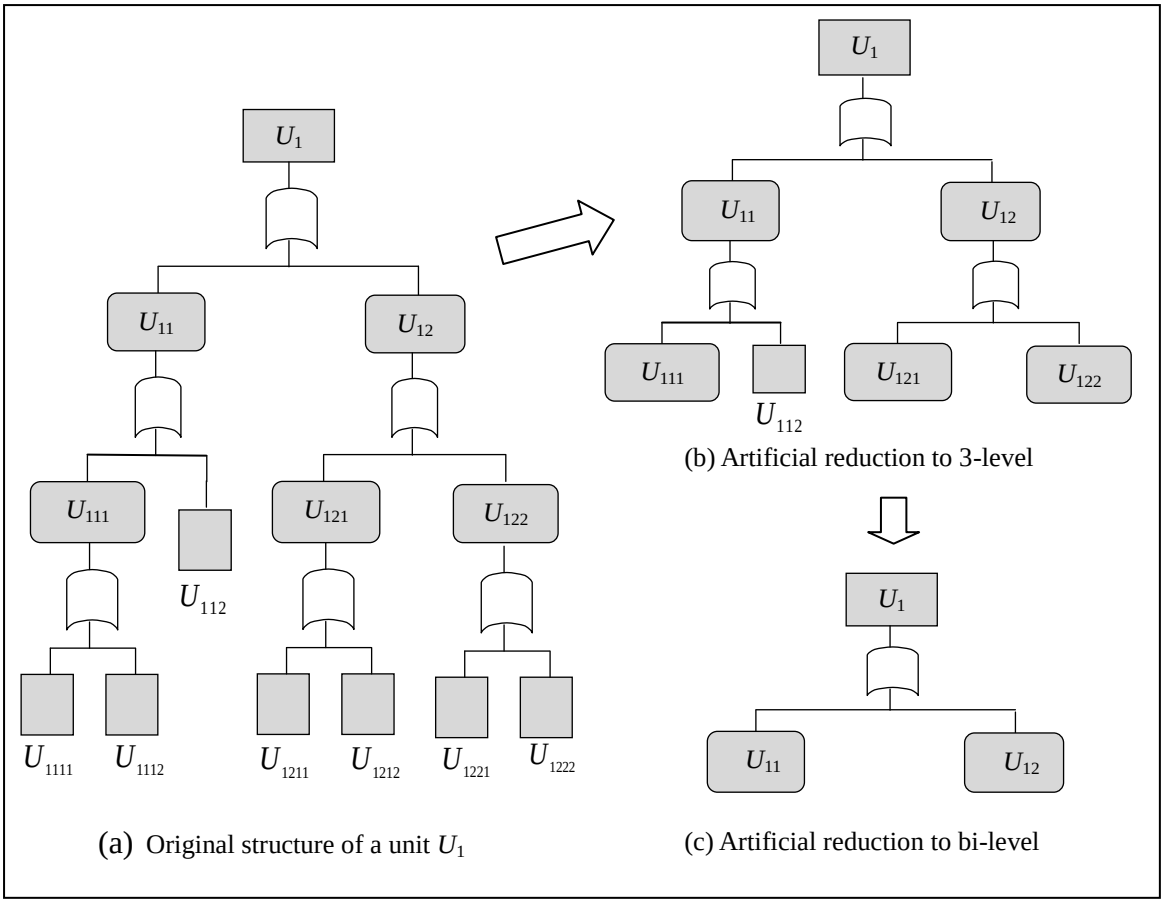


Fig. 2.10. Artificial reduction of hierarchical levels applied in traditional approach.

The conventional optimization approach more or less confined to component level. Therefore, multilevel redundancy allocation configuration has to be reduced to component level or bi-level configuration before applying traditional approach of optimization. This artificial shrinkage of configuration will yield suboptimal solutions which are undesirable. This research focuses on developing new metaheuristic methodology to represent and maintain original structure when solving MRAOPs which is not an easy task. In the next chapter, the proposed methodology, superior to existing approach, is presented and the effectiveness of this approach is demonstrated by solving numerical examples.

2.7 Summary

This chapter presented the concepts of hierarchy and modularity in system design. The hierarchy helps to simplify the design of large scale systems and provide decomposability

that helps to address the issue of managing the system more effectively throughout the life cycle. The structure of RBD plays a very important role in optimizing the reliability design of a system. The hierarchical concept of RBD is proposed in this chapter. This will help to simplify the design of complex system and represent exactly all the logical relationship between its subsystems and components.

The superiority of modular design is hard to challenge. This chapter described the modular redundancy concepts in optimal reliability design. The practical significance of the modular redundancy allocation in making a system more fault tolerant when so optimizing hierarchical RBD is explained. Finally this chapter proposed a general formulation of multilevel redundancy allocation optimization problems. The proposed formulation has several novelties. This formulation allows redundancy allocation to all units at every level. Bi-level series and parallel modules is proposed as building blocks to represent all possible hierarchical RBD. Modular redundancy allocation can easily be applied when optimizing such hierarchical RBD. Moreover, the proposed formulation achieves not only modular or component allocation but also mixed redundancy allocation that allows redundancy allocation at more than two levels simultaneously. This scheme is a paradigm shift in RBD representation for optimal reliability design of large scale systems and has a potential to provide solutions very close to global optimal solutions.

Chapter 3

Multilevel Redundancy Allocation Optimization using Hierarchical Genetic Algorithms

3.1 Introduction

Almost all of the large scale system exhibit hierarchical configurations multiple level of hierarchy within these configurations. Typical systems contain multiple levels, with the entire system at the top level, subsystems at lower levels inside the system, and down to the components at the lowest levels inside the various subsystems. Hierarchical systems such as these are termed multilevel systems, and their reliability depends on the reliability values of lower subsystems. For example, if the lower subsystems of a bi-level system are connected serially, the system reliability is the product of the reliability values of the lower subsystems. Fig.3.1 illustrates a schematic diagram of the multilevel configuration of a hierarchical reliability block diagram in a hierarchical product design.

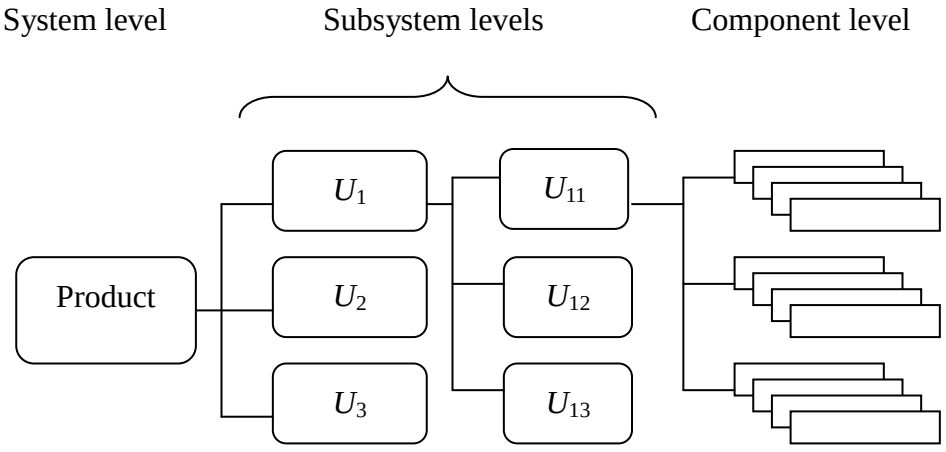


Fig. 3.1. A multilevel RBD.

The system reliability of a multilevel design configuration is usually optimized by allocating appropriate redundancy to less reliable subsystems or components at different levels, subject to certain constraints. This optimization technique is called multilevel

redundancy allocation optimization (MRAO), and subsequently formulated problems are called multilevel redundancy allocation optimization problems (MRAOP).

MRAOP are particularly attractive because real world systems and products are increasingly complex, and the system reliability of the multilevel configurations of these complex designs can be significantly improved by using multilevel redundancy allocation techniques. Multilevel redundant designs are increasingly prevalent in many practical systems, such as communication systems, computing systems, control systems, and critical power systems[138]. Techniques for implementing redundancy span a wide spectrum in the design space, and can create high reliability systems. Moreover, recent progress in miniaturization has made it easier to provide redundancy at all levels, ranging from the system level down to component levels. This approach can boost system reliability remarkably because redundancy can be distributed to any component at any level without structural constraints.

The optimization of system reliability using multilevel redundancy allocation is widely practiced in industry. Most integrated memory circuits, and VLSI chips that include internal memory blocks, currently use a hierarchical redundancy allocation scheme to enhance reliability, and chip yields. Also, a significant advantage of multilevel or hierarchical allocation is that it permits a modular scheme of redundancy allocation. Koren et al.[139] described how such modular schemes are particularly applicable when designing fault-tolerant or self-repairing semiconductor devices. Multilevel architectures that provide physical protection are now commonly implemented to increase the survivability of real systems in adverse conditions[140]. For protecting archived data, multilevel redundant designs in redundant arrays of inexpensive disks (RAID) that provide fault tolerance against disk failures outperform other RAID designs[141]. Several examples of multilevel RBD structures can be found in the literature, such as hierarchical series, hierarchical series

parallel, and others[142][1].

Almost all previous research in redundancy allocation optimization problems has focused on system configurations such as series-parallel , parallel-series , general networks, *k-out-of-n*: G(F), and other unspecified configurations, classified by Tillman, Hwang, & Kuo[3]. Kuo & Zuo provided good details concerning optimal reliability modeling[143], and the review paper by Kuo & Prasad[6] presents an overview of system reliability optimization. However, a comprehensive examination of this literature reveals that multilevel redundancy allocation problems are seldom addressed in terms of the detailed modeling or appropriate optimization techniques that such problems actually require. Also, attention paid to redundancy allocation is mainly confined to a single level, principally due to the notion that single-level redundancy yields better system reliability. We feel that this is not always the case. Boland & EL-Neweihi[144] demonstrated that this result does not hold in cases of redundancy configurations using non-identical parts.

According to Chern, redundancy allocation optimization problems are nonlinear integer programming problems, and NP-hard[2]. Besides being NP-hard, MRAOP qualify as hierarchical optimization problems[145]. The optimization of such hierarchical optimization problems beyond two levels, however, is more difficult using heuristics or exact algorithms. This is because multilevel allocation optimization problems generate a very large search space, and searching for optimal solutions using exact methods or heuristics will necessarily be extremely time consuming. Therefore, metaheuristic algorithms, particularly genetic algorithm (GA), are suitable for solving the multilevel redundancy allocation optimization. The seminal work by Goldberg[122] demonstrated that GA are very useful for solving complex discrete optimization problems, and the multiple solutions that GA provide allow considerable, valuable flexibility when choosing the best solution. This is one reason that GA is popularly applied to a variety of reliability optimization

problems[146-151][20]. However, none of the above-cited research specifically aims to optimize system reliability beyond two-level systems, and their subsystems.

Recently, the growing research interest in multilevel reliability modeling, and multilevel optimization using GA is reflected in the literature, due to the practical importance of these techniques. Levitin[140] proposed an algorithm for solving multilevel protection cost minimization problems subject to survivability constraints. This algorithm is based on a universal generating function technique used for system survivability evaluation, and on a genetic algorithm used as an optimization engine. Later, Yun & Kim[152] proposed a restricted multilevel redundancy allocation model, and optimized a three-level series redundancy allocation problem using a customized GA. However, this model allows redundancy allocation to only one unit at a given level in a direct line, which is defined as a set of units in which every unit except the system has a parent unit, and no other cousin units, the other units at the same level, are present in that set. Direct line concepts are explained by an example in a later section of this chapter. The purpose of using direct lines is to transform the multilevel design variables into vector representations, because conventional GA use one-dimensional representations of design variables. Unfortunately, the additional constraints imposed when transforming hierarchical design variables into vector design variables artificially constrict the feasible design region, often leading to suboptimal solutions.

Several genetic algorithms use a hierarchical approach to solve classes of hierarchical optimization problems. The hierarchical features offer the potential to address large problems efficiently[135]. De Jong et al[153] delineated classes of hierarchical problems, and described a framework for Hierarchical Genetic Algorithms (HGA), genetic algorithms that can exploit the structure present in hierarchical problems to achieve improvements in efficiency. These HGA exploit hierarchical features in different ways depending on the

problem, such as the use of a fitness-based hierarchy of populations[154], problem-specific subdivision of an algorithm into multiple levels[155], and the use of hierarchical representation by using control genes that regulate other genes[156]. Sefrioui & Periaux[157] developed HGA in which they used a hierarchical topology for the layout of sub-populations, achieving higher efficiency than conventional GA. Further, Yoshimura & Izui[158] proposed a genetic algorithm in which hierarchical genotype coding representation is used to exactly express the internal structure, and related hierarchical details. New crossover and mutation operators have been developed to handle these hierarchical genotypes during optimization processes.

The genotype coding representation used in the genetic algorithms proposed by Yoshimura & Izui[158] aims to represent the hierarchical design variables in design optimization problems for mechanical structures. However, the MROAP require a problem specific coding method for handling the logical linkages among the hierarchical design variables, and thus the coding scheme proposed by them cannot be applicable directly to solving MRAOP. Therefore, this research proposes a new variable coding method for the HGA first proposed by Yoshimura & Izui[158]. In this coding method, the phenotypes of hierarchical design variables are coded using two newly designed hierarchical nodal genotypes: the ordinal, and the terminal. These two nodal genotypes can be used as building blocks to codify most of the MRAOP hierarchical configurations. Thus, there is no need to transform the hierarchical design variables because these nodal genotypes preserve the exact hierarchical relationships within each design variable. The novelty of these hierarchical nodal genotypes is that they can express every possible combination of multilevel redundancy allocation, so that the optimization has a high probability of yielding nearly global optimal solutions.

The rest of the chapter is organized as follows. Section 3.2 describes the multilevel

series redundancy allocation optimization model. In Section 3.3, the HGA concepts are explained, and a HGA coding method for HS problems is proposed. In Section 3.4, we solve two series problems, a three-level problem, and a four-level problem. The optimal solutions obtained when using a conventional GA are compared with those obtained with the custom-coded HGA, and the resulting configurations are presented. Finally, the results are discussed in Section 3.5, while Section 3.6 concludes this chapter.

3.2 Multilevel redundancy allocation optimization problems

In a multilevel redundancy allocation model, each unit U_i can have x_i redundant units, and n_i sub-units, so there are $n_i x_i$ sub-units in the level below a parent unit. The sub-units n_i are different for each parent unit in the model described in chapter 2. Thus, the reliability R_i of unit U_i for multilevel series configurations can be calculated using (2.3) and (2.4). The multilevel reliability allocation model presented here allows redundancy for any unit at any level, and it is thus possible to achieve redundancy schemes that function at both the component, and modular levels. This mixed redundancy scheme allows the units to have redundancy not only at the same level, but also simultaneously for sub-units at lower levels.

The cost constraint of a multilevel redundancy allocation model also reveals hierarchical relationships among the multilevel units. The system cost is essentially the sum of the cost of subsystems and modules, and the cost of a module is the sum of all modules or component costs therein, when there are parallel units at the immediate lower level. In practical systems, it is assumed that multilevel redundancy incurs additional cost due to the adding or duplication of redundant units to modules, and the increased number of components. In general, the redundancy cost of R_i can be expressed mathematically as

$$C_i = \sum_{m=1}^{n_i} \sum_{j=1}^{x_i} C_{i,m}^{j} \times x_i + additional\ costs \tag{3.1}$$

Note that there are definite advantages to using modular redundancy in multilevel redundancy allocation, because the cost of adding, duplicating, or repairing a module is lower than carrying out a similar action upon a component. This result holds because the lower the level in a system, the more costly the repair job.

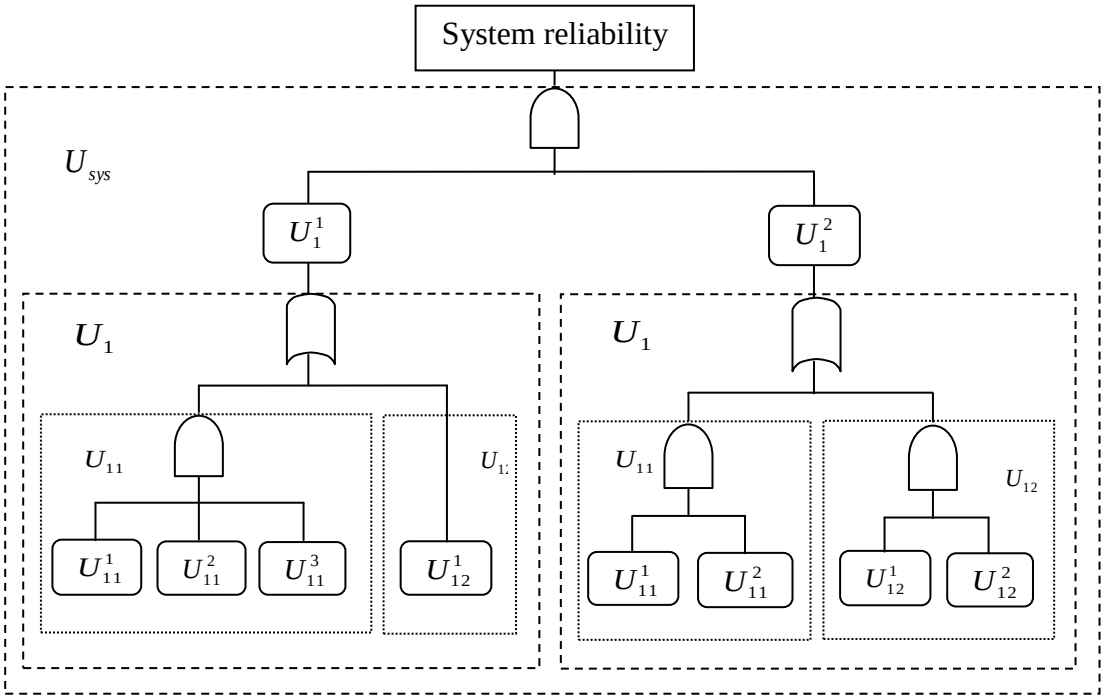


Fig. 3.2. An example of redundancy allocation in a unit U_1

The redundancy allocation optimization problem in a reliability system consisting of a set of design variables is expressed as

$$\text{Maximize } R_s = f(\boldsymbol{x}) \tag{3.2}$$

$$\text{Subject to C (x)} \le C_0 \tag{3.3}$$

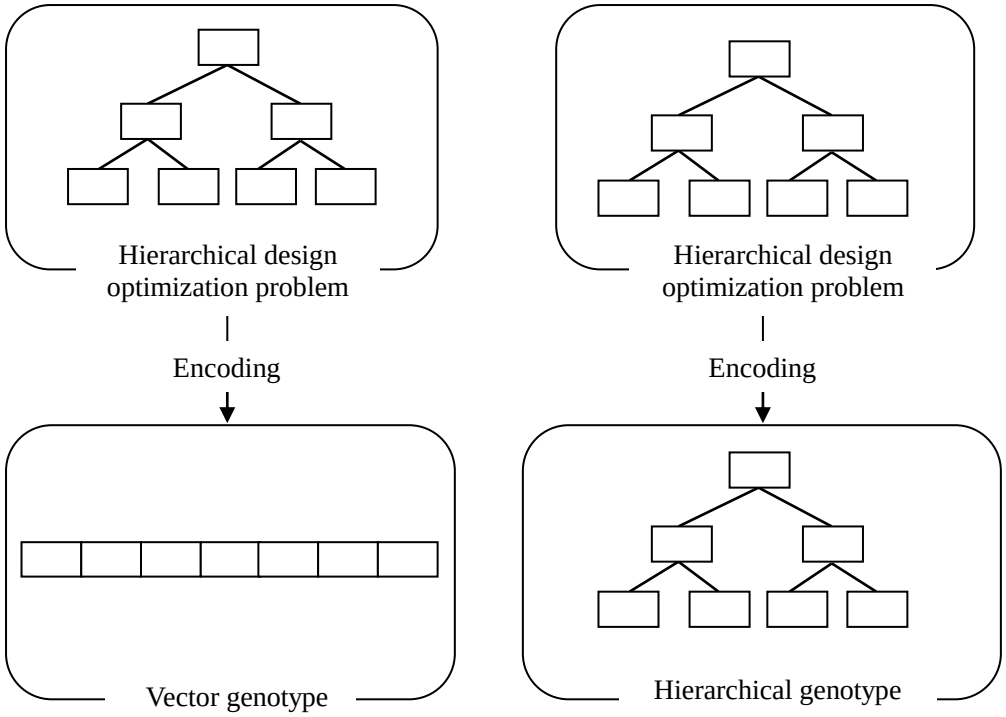
In a set of design variables $\boldsymbol{x}$, each design variable has a minimum, and maximum redundancy value. C_0 is a given, fixed positive value for the cost constraint. For example, the problem of optimizing a 2-level series redundancy allocation, as shown in Fig.3.2, can be stated mathematically as

$$R_{sys} = \left[1-(1-\{(1-\prod_{j=1}^{3}(1-R_{11}^{j}))(R_{12}^{1})\})(1-\{(1-\prod_{j=1}^{3}(1-R_{11}^{j}))(1-\prod_{j=1}^{2}(1-R_{12}^{j}))\}) \right] \quad (3.4)$$

The cost function used in this chapter for the cost constraint is described by (3.1).

3.3 Hierarchical genetic algorithms

A HGA[158] is an advanced genetic algorithm that can represent hierarchical and constraint relationships among design variables using hierarchical genotypes, and can optimize hierarchical problems in a single optimization process. This HGA is further customized with a new variable coding method, and subsequently applied to solve the MRAOP in this research. Fig.3.3 illustrates that conventional GA[122] use vector genotype structures, in contrast to HGA that use hierarchical genotype structures.



(a) Conventional GA (b) HGA

Fig. 3.3. Representation schemes of design variables in GA, and HGA.

The hierarchical redundancy allocation optimization problems here involve hierarchical relationships among design variables. Such hierarchical relationships can be handled well using hierarchical genotype representation. Because the HGA has special

types of genotype structures, new crossover, and mutation operators have to be applied. The HGA allows lower branches of the hierarchical structure to be exchanged, in addition to the exchange of genes. Using such genetic operations, new individuals are produced, and optimal structures can then be obtained.

3.3.1 Solution Encoding

A hierarchical genotype is represented using two types of nodes, ordinal, and terminal nodes, as shown in Table 3.1. Ordinal node N_i corresponds to redundancy unit U_i, and is characterized by several parameters, and design variables. Parameters k_i, and n_i stand for the redundancy of unit U_i, and the number of sub-units, respectively. Here, k_i is given by a design variable at an upper node, while the parameter n_i is a fixed value that depends on the optimization problems to be solved. $x_{i,m}^{j}$ is a design variable denoting the redundancy for the *m*-th sub-unit of the *j*-th redundancy unit, where j varies from 1 to k. Therefore, there are $n_i k_i$ design variables in unit U_i. A terminal node N_{t_i} corresponds to one of the lowest units, and incorporates design variable k_i, unit reliability r_i, and unit cost c_i. Because there are no sub-units, this terminal node does not contain parameter n_i, or design variable $x_{i,m}^{j}$. Using these two genotypes, all possible optimal solutions for series reliability allocation problems can be represented.

Furthermore, the ordinal, and the terminal genotypes each have two functions, namely, reliability, and cost. When the reliability function in the ordinal genotype is called, a calculation is conducted using (2.3). The particular equation selected depends on whether the unit is in series, or in parallel. When calculating either of these two equations, the reliability values of the lower units, $R_{i,m}^{j}$, are required; and these are obtained by calling the reliability function of the lower units. Finally, the reliability function of the terminal genotype returns its unit reliability r_i. Thus, the reliability functions are recursively called,

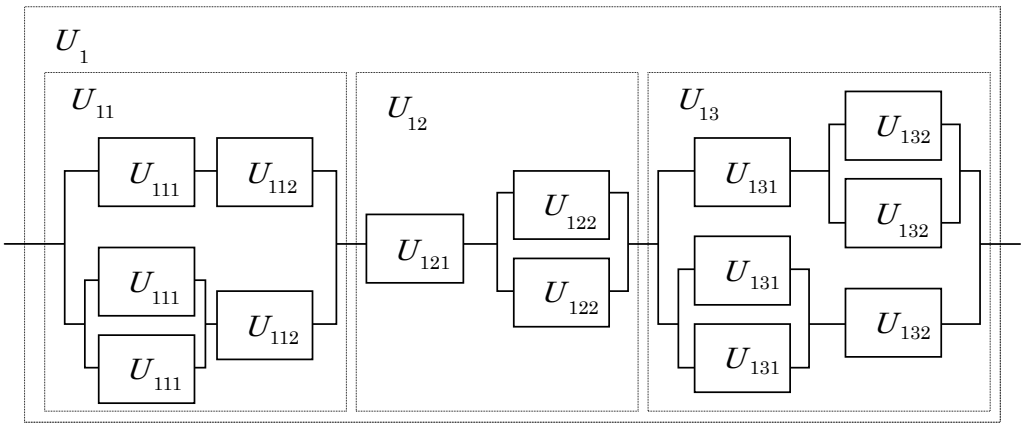
and the system reliability can be obtained. Similarly, the system cost can be obtained by calling the cost function embedded in each genotype.

TABLE 3.1

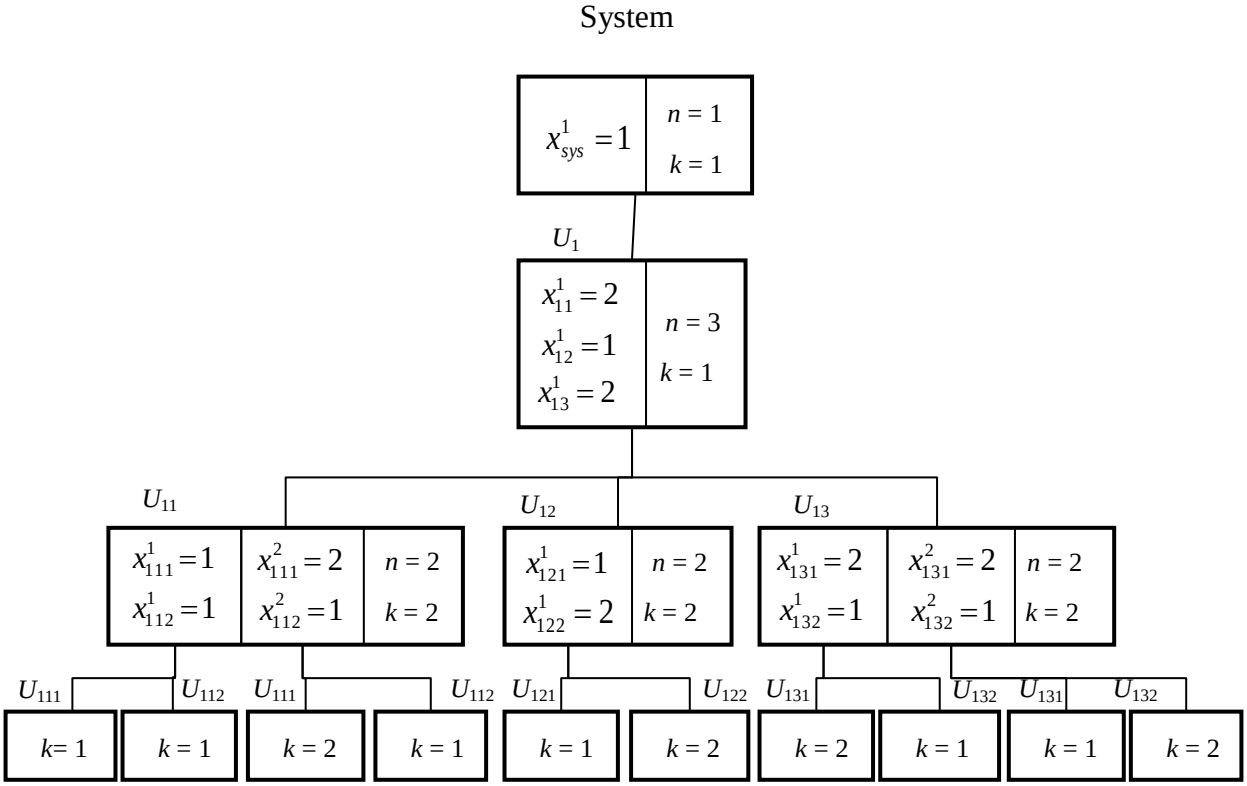
HIERARCHICAL GENOTYPE REPRESENTATION FOR SERIES SYSTEM

	Ordinal genotype node N_i	Terminal genotype node N_{t_i}
Design variable	$x_{i,m}^{j}$: the number of sub-unit for the m-th unit	
Parameter	k_i : the redundancy for unit U_i n_i : the number of sub-unit	k_i : the redundancy for unit U_i r_i : unit reliability c_i : unit cost

Fig.3.4 illustrates an example of the genotype encoding for a three-level series redundancy configuration. Fig.3.4(a) shows an optimal redundancy configuration for a system U_1 consisting three modules, U_{11}, U_{12}, and U_{13}, at the second level. The ordinal, and terminal nodes are assigned to represent modules, and component units at each level. Note that unit features, such as the redundancy and configuration, series or parallel, are expressed in the corresponding upper node.



(a) An example of a multilevel reliability system U_1.



(b) Design variable values at each ordinal, and terminal node.

Fig. 3.4. Hierarchical genotype representation in system U_1.

The HGA example shown in Fig.3.4(b) illustrates that genotypes using fixed arrays, which are frequently used in various optimization problems, are not applicable to this problem because the number of design variables varies according to the number of redundant units. In other words, the number of genes varies dynamically based on the proposed solution configuration. In this case, the two design variables, x^1_{111}, and x^2_{111}, represent the redundancy of U_{111}, because there are two redundant units for U_{11}, which is the unit above U_{111} in the hierarchy. If the number of redundant units for U_{11} increases, the number of design variables for U_{111} will also increase. The solution encoding scheme proposed in this research can successfully represent different numbers of design variables at every hierarchical level.

3.3.2 Objective function

A penalty function method has been applied to transform the constrained problem into an unconstrained problem, by penalizing infeasible solutions via a penalty term added to the objective function for any violation of the constraints. In this research, we used Gen & Cheng's method[159], which applies a severe penalty to infeasible solutions. The fitness function, $eval(\boldsymbol{x})$, is calculated using

$$eval(\boldsymbol{x}) = f(\boldsymbol{x}) \times p(\boldsymbol{x}) \tag{3.5}$$

where, $f(\boldsymbol{x})$, $p(\boldsymbol{x})$, and $\boldsymbol{x}$ are the system reliability, penalty function, and a set of design variables, respectively. We calculate the value of $p(\boldsymbol{x})$ using Gen & Cheng's penalty function for each individual; and for highly constrained optimization problems, the infeasible solutions occupy relatively large portions of the population at each generation. The penalty approach here adjusts the ratio of penalties adaptively at each generation to achieve a balance between the preservation of information, the selective pressure for infeasibility, and the avoidance of excessive penalization.

3.3.3 Crossover

Crossover operations between individuals are conducted among each corresponding set of genes, using a two-step procedure. For the initial step, any other individual is first selected as the crossover partner, and crossover operators then exchange the corresponding genes of the two individuals. Here, when a gene of an alternative for a substructure is exchanged with the corresponding gene of another alternative, all corresponding lower substructures are also exchanged, to preserve consistency in the selection of alternatives. If this operation were not conducted in this way, meaningless lower structures might be generated in the lower positions of the exchanged substructures. The algorithmic procedures are as follows.

Step 1 Select two individuals for crossover operations, then find the set of genes at the

highest level of the multilevel structural system for each of the two individuals, and start the crossover operation with probability p_{c_1}.

Step 2.1 If the gene $x_{i,m}^{j}$ of individual 1, and that of individual 2, are different, then conduct a crossover operation for $x_{i,m}^{j}$ with probability p_{c_2}. This operation is the same as a uniform crossover of simple genetic algorithms with p_{c_2} set to 0.5. Then, proceed to Step 2.3. If crossover operations are not conducted, proceed to Step 2.4. If the genes of both individuals are the same, proceed to Step 2.2.

Step 2.2 If $x_{i,m}^{j}$ contains a subordinate set of genes, it will be examined for possible crossover operations in Step 2.1. Otherwise, proceed to Step 2.4.

Step 2.3 When $x_{i,m}^{j}$ genes are exchanged between individuals 1 and 2, the lower substructures of each individual are also exchanged.

Step 2.4 Increment m by 1. When $m > n$, set m =1, and increment j by 1. When $j > k_i$, end the crossover operations because the set of genes has been exhausted, and return to the crossover operations for the parent set of genes.

3.3.4 Mutation

In mutation operations, mutation operators are first applied to the set of genes at the highest level of the multilevel structural system, and mutation operators are recursively applied to their child sets of genes in the same way as for crossover operators. The algorithmic procedures are as follows.

Step 1 Examine the substructure at the highest level.

Step 2.1 Determine whether or not a mutation operation should be conducted, with mutation probability p_m for the gene $x_{i,m}^{j}$. If the mutation is conducted,

proceed to Step 2.3. Otherwise, proceed to Step 2.2.

Step 2.2 If $x_{i,m}^{j}$ contains a child set of genes, proceed to Step 2.1, and examine the child set of genes. If not, proceed to Step 2.5.

Step 2.3 Randomly generate $x_{i,m}^{j}$.

Step 2.4 Randomly reconstruct the genes of all sub-units for the selected alternative.

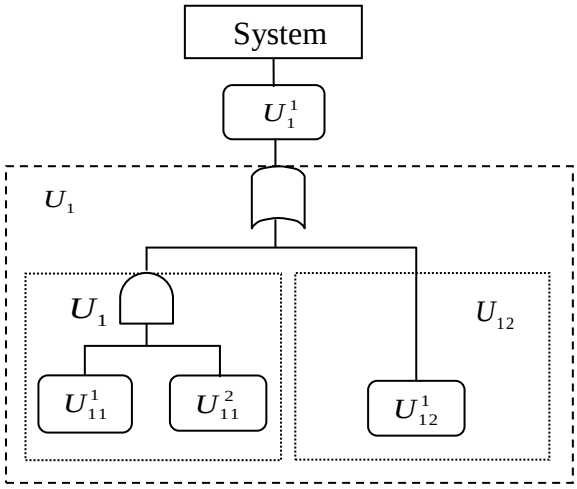
Step 2.5 Increment m by 1. When $m > n$, set m =1, and increment j by 1. When $j > k_i$, end the crossover operations because the set of genes has been exhausted, and return to the crossover operations for the parent set of genes.

3.4 Conventional genetic algorithm

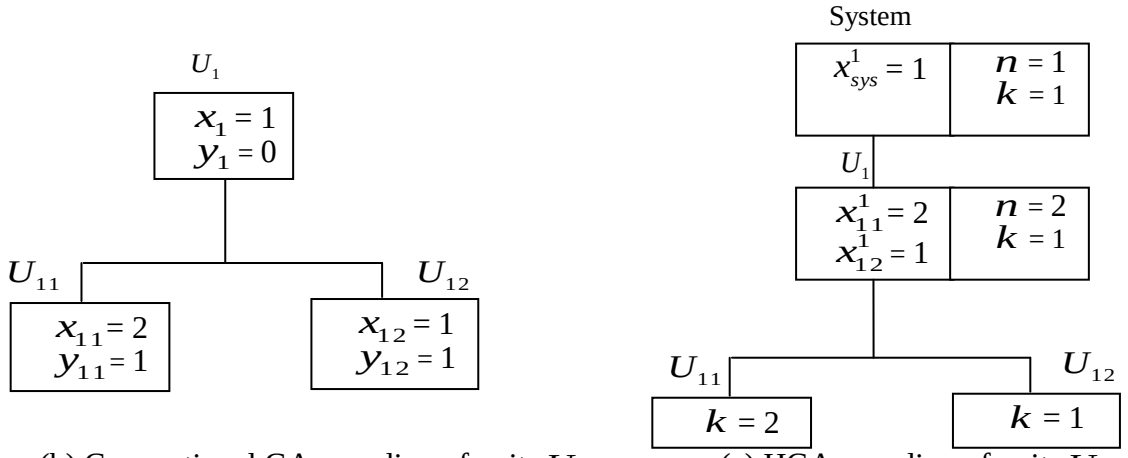
We applied the GA, proposed by Yum & Kim[152] to solve MRAOP to compare the obtained solutions with those obtained by the HGA. We call this GA a conventional GA because it uses vector coding of the design variables, and applies a special crossover & mutation operator to handle such coding. The genotypes for the conventional GA[152] are encoded as an ordered couple of a design variable, $x_{i,m}$, and an indicator variable, $y_{i,m}$; v_i =($x_{i,m}$, $y_{i,m}$), where the subscript *i* is the index of the chromosome to which the gene belongs, and subscript *m* denotes units. A chromosome is represented as

$$v_i = [(x_{i1}, y_{i1})(x_{i2}, y_{i2})...(x_{in}, y_{in})]$$

The value of the indicator variables for a unit is 1 when that unit is subject to redundancy, and 0 when that unit is not allowed to have redundancy. Only one unit among the set of units in a direct line is selected to have redundancy so that the sum of the indicator variables of units along a direct line is 1. On the other hand, we used hierarchical genotype encoding when applying the HGA to solve the MRAOP.



(a) Redundancy allocation in unit U_1 .



(b) Conventional GA encoding of unit U_1 . (c) HGA encoding of unit U_1 .

Fig. 3.5. Coding schemes in conventional GA, and HGA for a series system U_1 .

Coding schemes for conventional GA, and HGA can be understood more clearly by examining an example of redundancy allocation in a bi-level series unit U_1 having two sub-units, U_{11}, and U_{12}, as shown in Fig.3.5. The redundancy values for sub-units U_{11}, and U_{12} are 2, and 1, respectively. Note that there are two direct lines, (U_1-U_{11}), and (U_1-U_{12}) in Fig.3.5(b). Because only a unit at a level is selected to have redundancy among the set of units in a direct line, unit U_1 cannot have redundancy if units U_{11} and U_{12} are subject to redundancy. Thus, the GA coding scheme does not allow redundancy at two levels simultaneously. In contrast, the HGA allows redundancy at two levels simultaneously.

Fig.3.5(c) shows the reliability for both the system, and the sub-units. Both a conventional GA, and a HGA were applied when optimizing multilevel series redundancy allocation problems with different configurations, to evaluate their applicability for solving multilevel allocation problems. The cost function $C(x) = cx + \lambda^x$ is used as a constraint. The symbols x , cx, and λ respectively represent the number of parallel units, the unit cost, and the additional cost.

3.5 Numerical Examples

The HGA was applied to optimize multilevel series redundancy allocation problems having two different configurations. The first configuration is called *problem-A*, and is similar to the problem described in Yum & Kim[152], while the second configuration is called *problem-B*. Fig.3.6, and Fig.3.7 respectively represent *problem-A*, and *problem-B*. *Problem-A* contains three levels, and *problem-B* contains four levels. All units of these configurations are in series.

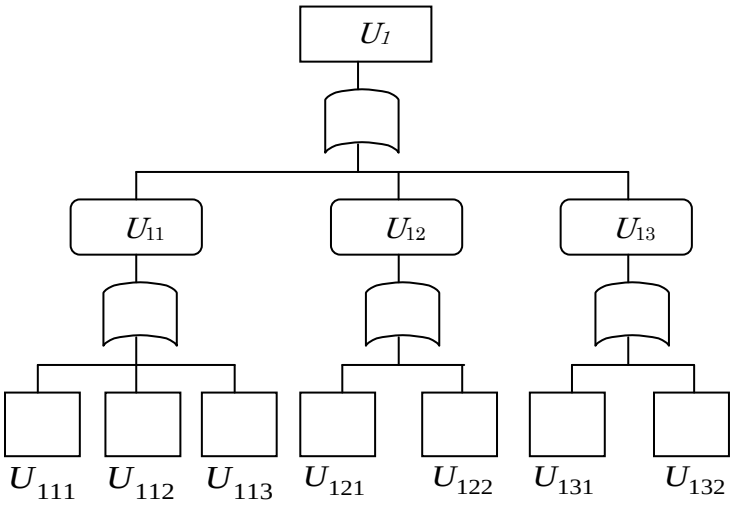


Fig. 3.6. *Problem-A* (a three level MS system).

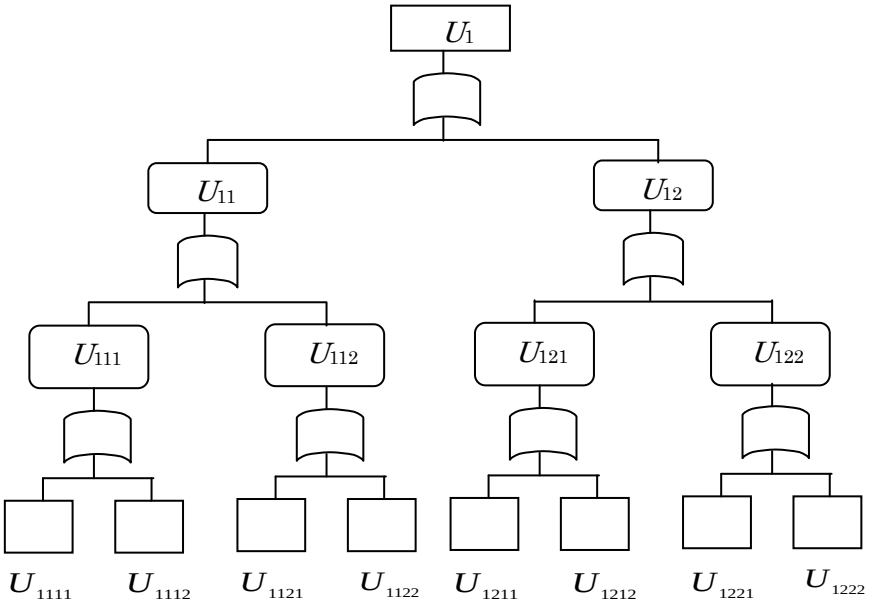


Fig. 3.7. *Problem-B* (a four level MS system).

3.5.1 Input data

Suitable parameters for optimizing the two allocation problems were selected based on several experimental runs using a conventional GA, and the HGA we created. We observed the convergence of fitness functions, and selected suitable GA operator values for subsequent use in the optimization process. Table 3.2 provides a summary of the average, and best fitness values for different HGA parameters obtained during 20 runs with 500 generations in each run. The best crossover, and mutation rate values for solving these problems when using a conventional GA were 0.8, and 0.1, respectively. Similarly, when using the HGA, these best values were respectively 0.8, and 0.05. An initial population of 100 individuals was generated randomly when using both the GA, and HGA. This population size was selected based on the performance evaluation of the algorithms with different population sizes.

TABLE 3.2

HGA PARAMETERS

Cases	Parameters		Average Fitness	Best Fitness
	Crossover	Mutation	(20-runs)	(20-runs)
1	0.7	0.05	0.96047	0.97628
2	0.9	0.05	0.96155	0.97628
3	0.8	0.05	0.9621	0.97639
4	1.0	0.05	0.96117	0.97628
5	0.8	0.01	0.92826	0.97254
6	0.8	0.10	0.94484	0.96422
7	0.8	0.20	0.93190	0.95082

TABLE 3.3

INPUT DATA

Problem-A[152]				*Problem-B*			
Unit	Reliability	Cost	λ	Unit	Reliability	Cost	λ
U_1	0.4003	72	2	U_1	0.2198	102	2
U_{11}	0.7267	26	2	U_{11}	0.5130	48	2
U_{12}	0.7650	19	3	U_{12}	0.4284	50	2
U_{13}	0.7200	21	2	U_{111}	0.7200	21	3
U_{111}	0.9000	5	3	U_{112}	0.7125	21	3
U_{112}	0.9500	6	4	U_{121}	0.6300	23	3
U_{113}	0.8500	5	3	U_{122}	0.6800	21	3
U_{121}	0.9000	6	4	U_{1111}	0.9000	7	4
U_{122}	0.8500	7	4	U_{1112}	0.8000	6	4
U_{131}	0.9000	8	3	U_{1121}	0.7500	8	4
U_{132}	0.8000	7	4	U_{1122}	0.9500	5	4
				U_{1211}	0.7000	9	4
				U_{1212}	0.9000	6	4
				U_{1221}	0.8500	5	4
				U_{1222}	0.8000	8	4

Twenty two design variables were used with the conventional GA, which is the sum of the redundancy numbers plus the constraints for direct lines. In contrast, the number of design variables used with the HGA was 11. The number of generations was 500 in each

case, and a maximum redundancy number of five was imposed for both the modular, and component redundancy schemes. The unit reliability, and the unit cost at the very lowest level in the multilevel redundancy allocation problems were used when calculating the unit reliability, and the unit cost of upper level units, up to the system level. Table 3.3 summarizes the unit reliability, and unit cost of the components at the very lowest level in both problems. Note that we used the same data for *problem-A* that Yum & Kim[152] used, to enable a comparison of the optimal solutions obtained by the HGA with those provided by a conventional GA.

3.5.2 Computational results

We separately applied the HGA, and GA when solving the *problem-A*, and *problem-B* allocation optimization problems. First, we checked the convergence of the optimal solutions when using the GA, and HGA; and Fig.3.8, and Fig. 3.9 show the results when using the two different types of algorithm. The x-axis represents the number of generations, and the y-axis represents the system reliability. The cost constraints for these two graphs were 240 for *problem-A*, and 500 for *problem-B*.

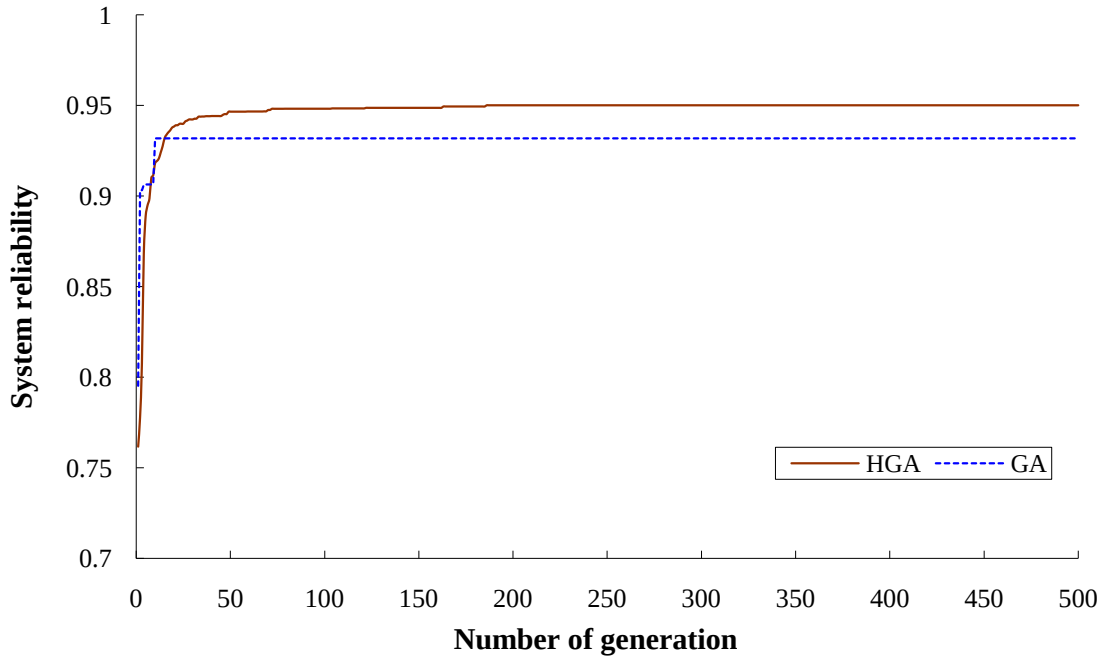


Fig. 3.8. Convergence of GA, and HGA in *problem-A*.

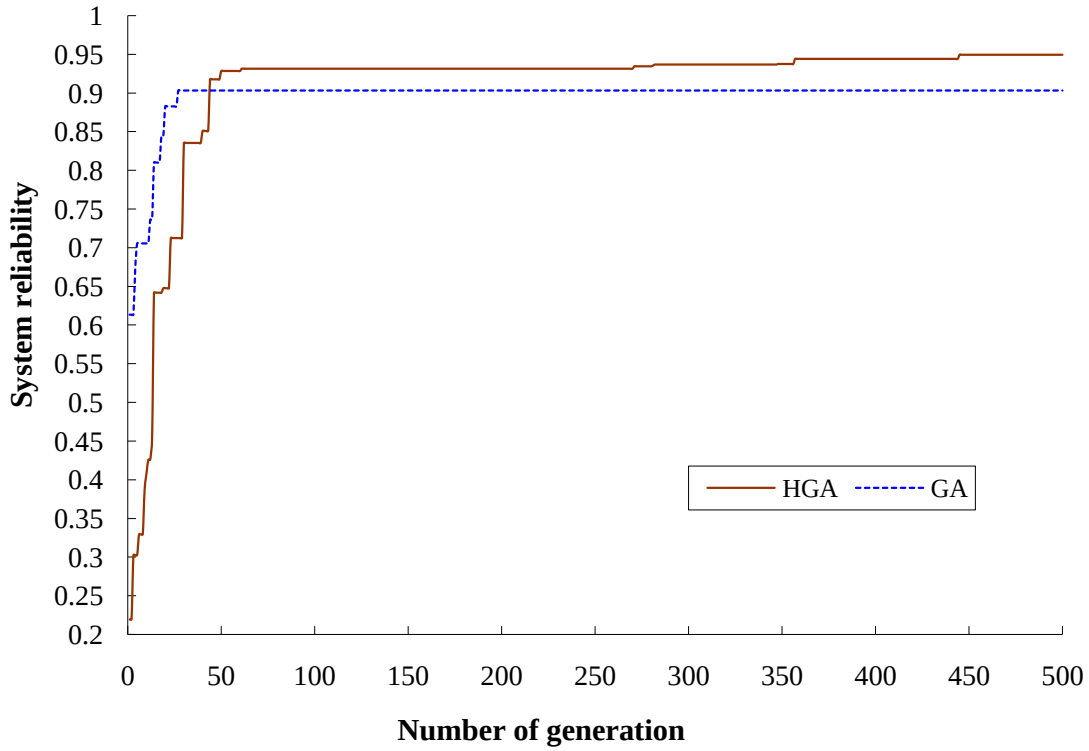


Fig. 3.9. Convergence of GA and HGA in *problem-B*

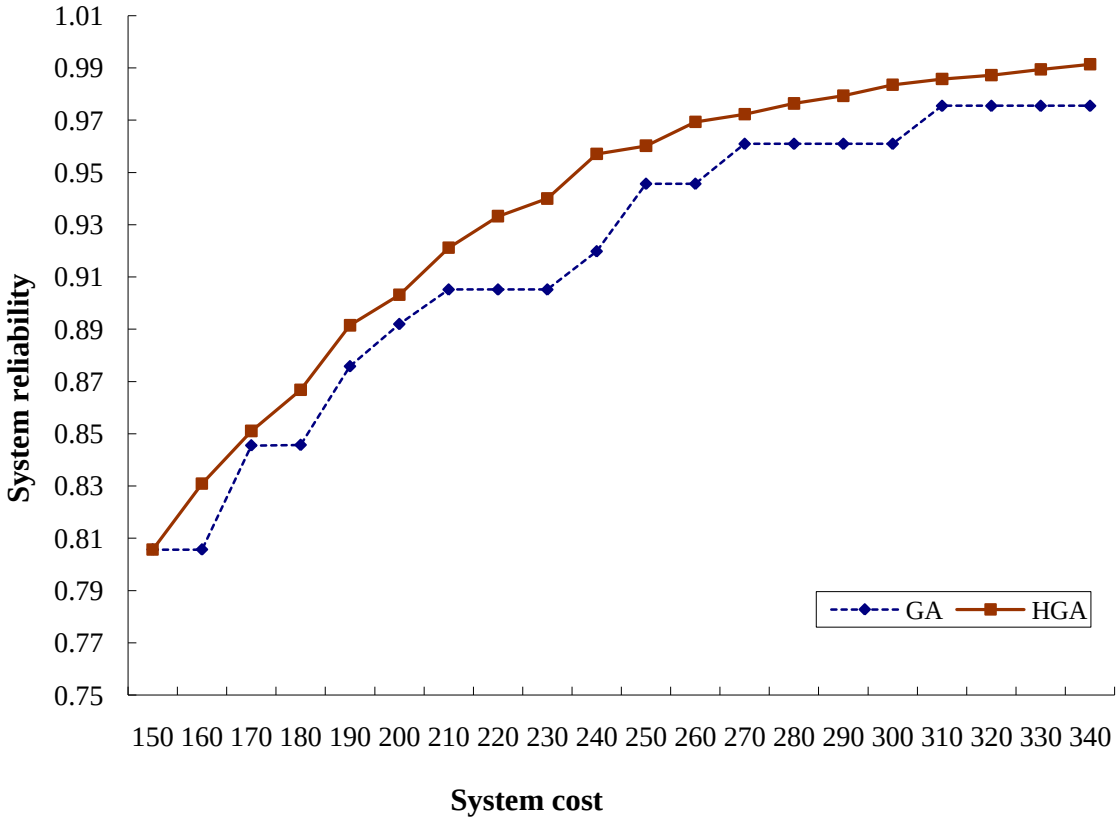


Fig. 3.10. Optimal solutions for *problem-A* obtained using GA, and HGA.

To assess the influence of cost constraints upon the optimal solutions, 20 cases for a 3-level problem, and 15 cases for a 4-level problem, were examined. Ten 500-generation trials were performed using each algorithm type, and the best solution of the ten-trial set was chosen as the optimal solution in each of these cases. Fig.3.10, and Fig.3.11 show the trends of optimal solutions obtained using the GA, and HGA. The x-axis represents the cost constraint, and the y-axis represents the optimal system reliability.

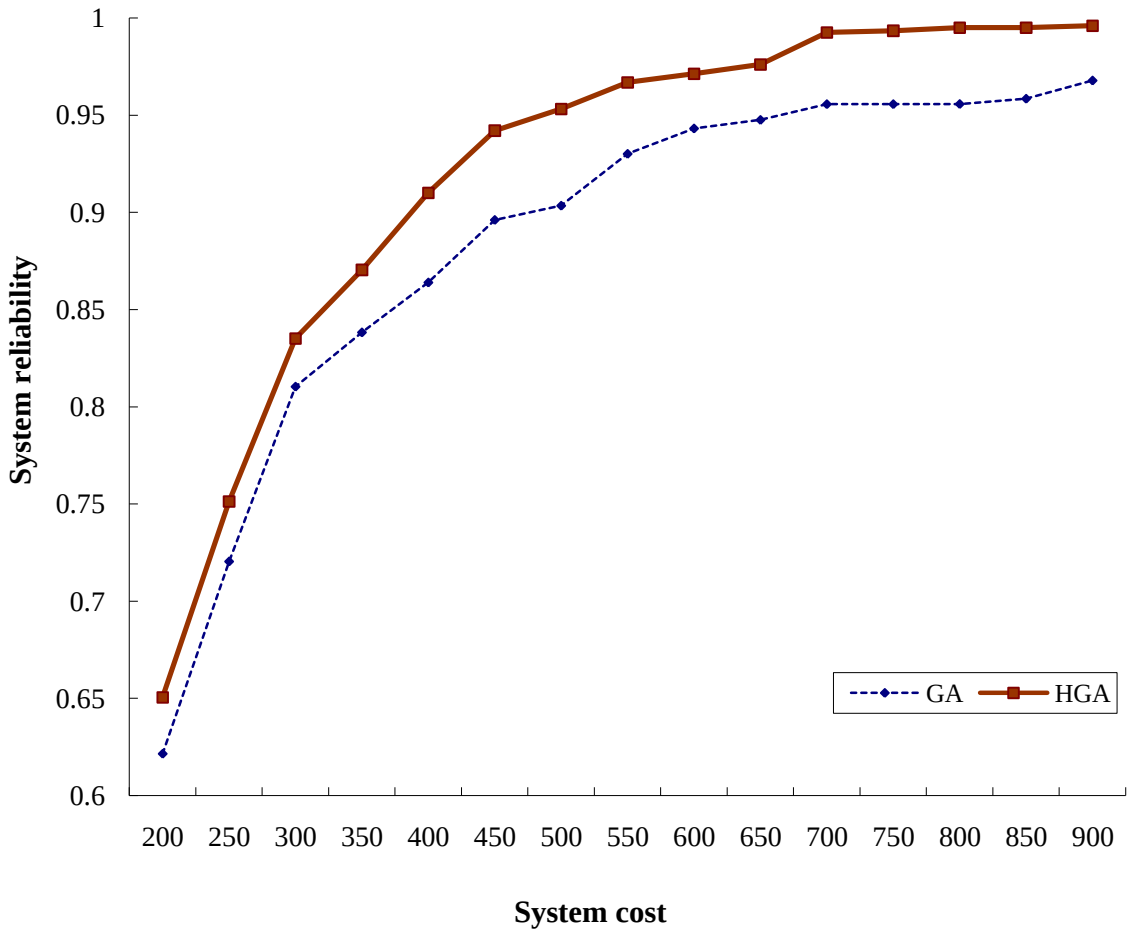


Fig. 3.11. Optimal solutions for *problem-B* obtained using GA, and HGA.

TABLE 3.4

OPTIMAL HIERARCHICAL CONFIGURATIONS (*PROBLEM-A*)

Problem-A						
Cases	GA			HGA		
	Reliability	Cost	Optimal Allocation $[x_1x_{11}x_{12}x_{13}x_{111}x_{112}x_{113}x_{121}x_{122}x_{131}x_{132}]$ $[y_1y_{11}y_{12}y_{13}y_{111}y_{112}y_{113}y_{121}y_{122}y_{131}y_{132}]$	Reliability	Cost	Optimal Allocation $[(x_1)(x_{11}x_{12}x_{13})(x_{111}x_{112}x_{113})(x_{121}x_{122})(x_{131}x_{132})]$
1	0.9276	289	[14333242224] [01110000000]	0.9742	290	[(1)(222) (211322)(2121)(1122)]
2	0.7822	278	[24241232212] [01010001100]	0.8537	289	[(1)(221) (222212)(1122)(32)]
3	0.9557	275	[54333343552] [01110000000]	0.9622	297	[(1)(122) (223)(2211)(2122)]
4	0.7989	278	[34542212225] [01010001100]	0.8734	292	[(1)(212) (122122)(22)(2222)]
5	0.8447	291	[45333232231] [01010001100]	0.9029	290	[(1)(221) (221221)(2122)(32)]
6	0.8506	292	[25321433131] [01110000000]	0.9102	286	[(1)(221) (322212)(2211)(22)]
7	0.8986	275	[34335144225] [01110000000]	0.9187	300	[(1)(212) (212212)(32)(1122)]
8	0.9272	298	[43243533234] [01010001100]	0.9579	294	[(1)(122) (322)(2222)(1212)]
9	0.9262	270	[53341311213] [01110000000]	0.9433	300	[(1)(132) (322)(121111)(2132)]
10	0.9185	278	[34544552214] [01010001100]	0.9467	297	[(1)(212) (221222)(22)(2222)]

TABLE 3.5

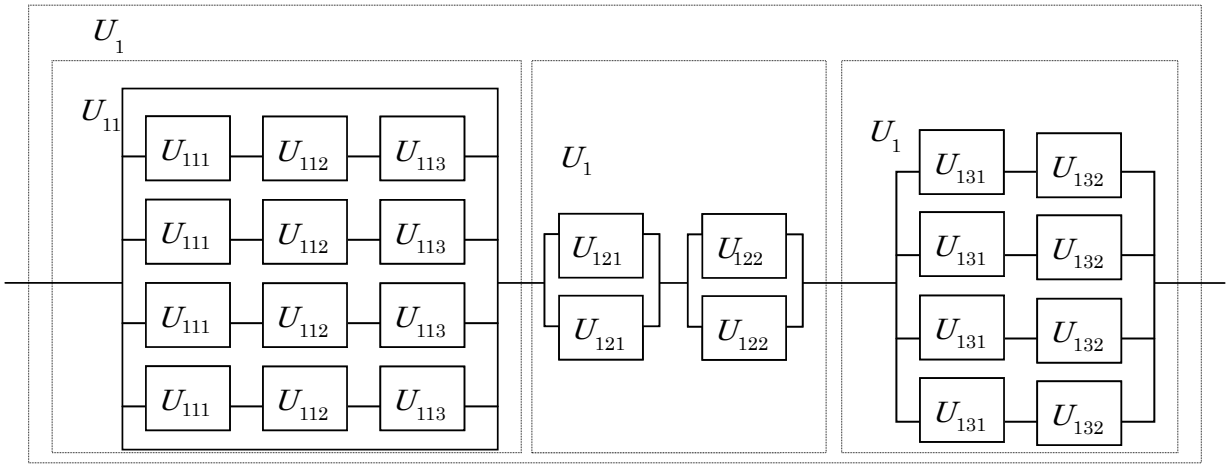
OPTIMAL HIERARCHICAL CONFIGURATIONS (*PROBLEM-B*)

Cases	*Problem-B*					
	GA			HGA		
	Reliability	Cost	Optimal Allocation $[x_1x_{11}x_{12}x_{111}x_{112}x_{121}x_{122}x_{1111}x_{1112}x_{1121}x_{1122}x_{1211}x_{1212}x_{1221}x_{1222}]$ $[y_1y_{11}y_{12}y_{111}y_{112}y_{121}y_{122}y_{1111}y_{1112}y_{1121}y_{1122}y_{1211}y_{1212}y_{1221}y_{1222}]$	Reliability	Cost	Optimal Allocation $[(x_1)(x_{11}x_{12})(x_{111}x_{112})(x_{121}x_{122})(x_{1111}x_{1112})(x_{1121}x_{1122})(x_{1211}x_{1212})(x_{1221}x_{1222})]$
1	0.9568	484	[155132223244513] [001010011000000]	0.9775	499	[(1)(22)(11)(21) (1222)(2222)(32) (33)]
2	0.7810	485	[143143432441532] [000011011000011]	0.8677	496	[(1)(11)(21)(12) (2212)(33)(23) (2222)]
3	0.9568	484	[155132223244513] [001010011000000]	0.9777	486	[(1)(11)(22)(12) (2212)(2122)(33) (2222)]
4	0.8202	486	[224335554332233] [000100000111111]	0.8870	460	[(1)(11)(12)(12) (32)(2223)(22) (2222)]
5	0.8586	485	[213431342143232] [000110000001111]	0.9368	454	[(1)(11)(12)(12) (32)(2223)(22) (2222)]
6	0.8767	462	[135443341233234] [000101100110000]	0.9508	491	[(1)(11)(32)(21) (112222)(2123)(2211) (32)]
7	0.9124	481	[253452354243211] [010000100001100]	0.9538	467	[(1)(11)(12)(22) (23)(1212)(2132)(2222)]
8	0.9515	486	[111324452224143] [000101100110000]	0.9741	491	[(1)(11)(12)(22) (23)(2122)(2232) (32212)]
9	0.9122	462	[153433255123132] [000111000000011]	0.95021	467	[(1)(11)(22)(21) (2222)(2222)(2222) (32)]
10	0.8941	441	[134433314313215] [000111100000000]	0.9645	494	[(1)(11)(23)(22)(2122)(112112)(2222) (2132)]

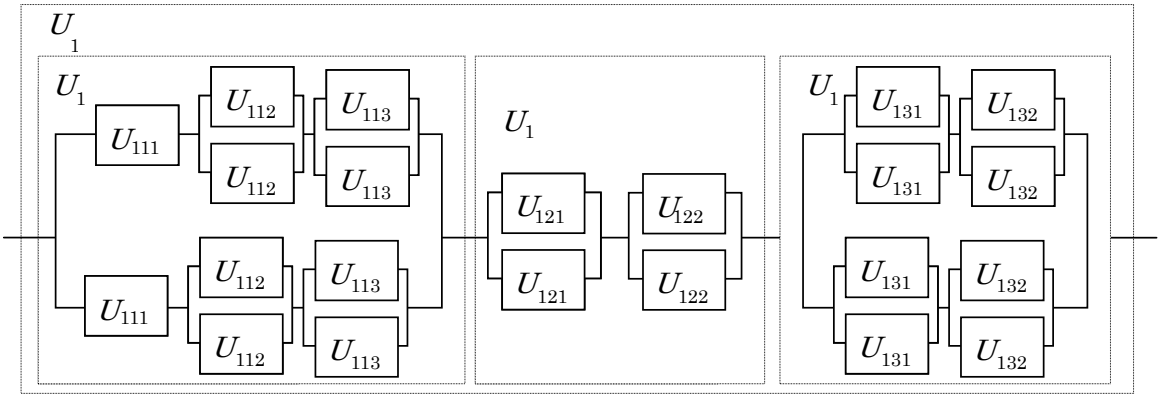
Next, we examined ten cases in which the unit reliability values were varied while the cost constraint was held to a value of 300 for *problem-A*, and 500 for *problem-B*. In the same manner as before, ten 500-generation trials for each of these ten cases were carried

out, and the best solution was chosen as the optimal solution for each case. Note that the number of function calls in each case considered here was the same for both GA, and HGA. Table 3.4, and Table 3.5 summarize the optimal solutions obtained when using the GA, and HGA for *problem-A*, and *problem-B*, respectively.

An interpretation of the optimal solution data summarized in these two tables is provided in Fig.3.12, which shows the arrangement of the units in *problem-A*, and *problem-B*. It is a graphic representation of the optimal solutions for the fourth case listed in Table 3.4.



(a) Optimal configuration obtained using GA.



(b) Optimal configuration obtained using HGA.

Fig. 3.12. Optimal solutions for the fourth case listed in Table 3.4.

Fig.3.12(a) illustrates the optimal arrangement of the modules and components in the

system obtained when using the GA, and Fig.3.12(b) illustrates the optimal arrangement obtained when using the HGA.

3.6 Discussion

The numerical examples solved in the previous section demonstrate that hierarchical genotype representations of hierarchical design variables provide superior solutions in comparison to vector representation. The most suitable GA, and HGA parameters were selected from the results of a number of preliminary runs; and Table 3.2 shows that the most useful HGA crossover, and mutation rates are 0.8, and 0.05, respectively, determined by twenty 500-generation runs. We observe in Fig.3.8, and Fig.3.9 that the HGA offers superior convergence, and that this advantage is achieved more smoothly by searching a larger feasible design space than when a conventional GA is used.

Moreover, Fig.3.10, and Fig.3.11 indicate that the optimal solution obtained using the HGA is superior to its conventional GA counterpart. After examining the solution data, we find that there is an approximately 4% maximum improvement in the 3-level series allocation problem, and a 5% improvement in the 4-level series allocation problem. Similarly, in Table 3.4, and Table 3.5, we see that the HGA yielded average improvements of 4.7%, and 5.82% over the conventional GA. Moreover, the maximum improvement in the optimal solutions when using the HGA was found to be 9.23% in the 3-level problem, and 11% in the 4-level problem. The improved reliability obtained using the HGA is achieved without incurring additional material or parts costs. This is an important milestone because, in high reliability applications, even very small improvements in reliability are often difficult to obtain. Thus, it appears incontrovertible that the hierarchical genotype scheme typical of HGA is better suited for optimizing multilevel allocation problems than the one-dimensional vector schemes of conventional GA.

The reason why the GA yielded inferior solutions in comparison to the HGA is that the

GA requires vector transformation of the hierarchical design variables. The vector transformation of hierarchical design variables into one dimensional array representations actually reduces the feasible design space, and the GA may consequently fail to find superior solutions that exist just beyond its feasible design space. Because HGA do not require vector transformation, the feasible design space remains unaffected, and this leads to better optimal solutions during the searching process. Additionally, the hierarchical coding method proposed in this research can express the exact internal structure with series linkage.

Furthermore, the simultaneous allocation of redundancy at two or more levels also leads to better solutions than those provided by conventional GA. Allocated resources can be appropriately shared at all levels, and one such optimal arrangement of redundant units is graphically illustrated in Fig.3.12. We see that the optimal HGA solution contains two parallel modules for unit U_{11}; and sub-units U_{111}, U_{112}, and U_{113} have single, double, and double redundancy, respectively. On the other hand, the optimal solution obtained using the conventional GA contains four parallel U_{11} modules, and all the sub-units have only single redundancy. A similar pattern can be seen concerning the other two U_1 modular units. Hence, an additional significant advantage that the use of HGA provides is that redundancy at both the unit, and the sub-unit level can be achieved simultaneously.

The performance of the HGA in solving the two examples here indicates that hierarchical genotype representation is not only capable of solving multilevel reliability optimization problems of any size, but also that it allows significant flexibility so that every possible redundancy combination can be evaluated. This flexibility in redundancy optimization seems impossible to achieve when using conventional GA. Another useful feature of hierarchical genotype representation is that optimal redundancies are given hierarchically for each module, and component. This is highly desirable in a complex

system, when the goal of ensuring optimum reliability depends on determining exactly how many redundancies are required for a particular module at a particular level in the hierarchical system.

3.7 Summary

Multilevel redundancy allocation optimization problems are frequently encountered in complex system designs. This chapter proposed a general formulation for multilevel redundancy allocation optimization problems that aim to maximize system reliability. These multilevel optimization problems have hierarchical design variables, so we proposed a new coding method for use in a HGA, in which hierarchical design variables of MRAOP are represented using two types of hierarchical genotype: nodal, and terminal. We applied the newly developed HGA, and a conventional GA separately, to solve two multilevel series redundancy allocation optimization problems having three, and four levels. The optimal solutions for these two problems demonstrated that the proposed HGA provides optimal system reliability that is superior to the conventional GA results, because it does not depend on the use of vector coding to represent the hierarchical variables, and can preserve the original design space. HGA using the new variable coding method presented here can be applied in other hierarchical optimization problems, but the efficiency of such algorithms must be investigated. We hope to extend our approach for optimizing the system reliability of other multilevel structures such as hierarchical series-parallel, multilevel network, and other multilevel configurations in future work.

Chapter 4

Optimal modular redundancy allocation in series and series-parallel systems

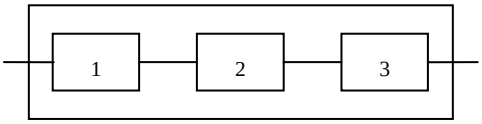
4.1 Introduction

Modularity in product design is a crucial topic when developing highly reliable product architectures. It is a key strategy for achieving better serviceability and reliability, particularly when designing products whose lifetime operational costs exceed the initial acquisition cost, such as for airplanes, locomotives, power generating plants, and major manufacturing equipment[160]. Most complex engineering systems of this kind contain thousands of different components that function interdependently, while certain components are used only for a specific set of subtasks within the system. Such sets of components having independent functions can be accommodated within a simple subsystem, or sub-unit. Here, such a subsystem is called a module. In system reliability theory, a module indicates a group of components that has a single input from, and a single output to, the rest of the system[143]. The contribution of all components in a module to the performance of the whole system can be represented by the state of the module. Once the state of the module is known, one does not need to know the states of the components within the module to determine the states of the system.

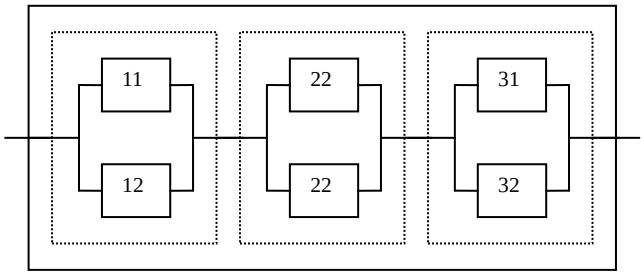
Systems that have modular subsystems usually have superior fault tolerance, ease of maintenance, and allow modules to be recovered for possible further use when the system as a whole has reached the end of its useful life[137]. Furthermore, a modular system is often simpler than a complex system built from single components. In essence, the modular architecture of a high-reliability design reduces the number of parts in an optimal configuration by providing a modular redundancy. Despite the subtle and profound benefits

of modular redundancy, which enhances fault tolerance and reduces lifecycle costs, optimizing modular-level allocation under resource constraints is a challenging task for design engineers.

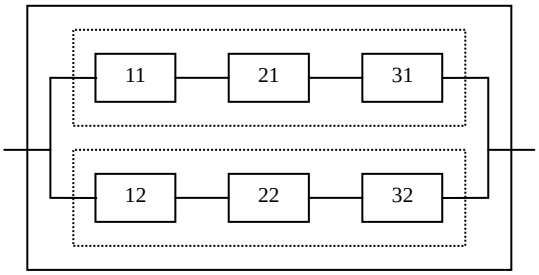
Conventionally, redundancy is added either to a component level or to a subsystem level, when optimizing system reliability. The redundancy added at the component level is termed component redundancy, and redundancy added at the modular level is termed modular redundancy. Specifically, a redundant module is a similar module added in parallel to the existing module to increase its reliability without altering its internal structure. Fig. 4.1 illustrates these two redundancy schemes in a series system containing three components.



(a) Basic reliability block diagram



(b) Component level allocation



(c) Modular level allocation

Fig. 4.1. Redundancy allocation in a series system containing three components

In other words, we preserve a module's internal structure, such as the arrangement of its sub-modules and components, while providing modular redundancy. Thus, we need not know the status of its components in order to know the status of the system. Modular redundancy therefore simplifies the complexity of the system and makes it easier to isolate faults in case of failure.

In the literature, we find proposals for various models that deal with several system

configurations, such as series, parallel, series-parallel, network, and *k-out-of-n* systems, and others. To maximize the system reliability of these models, a large number of techniques have been proposed for optimal redundancy allocation problems. Most techniques for redundancy optimization, however, have been limited to single levels[6]. Boland and EL-Neweihi[144] demonstrated that redundancy at the component level is not always more effective than redundancy at the system level for redundancy cases using non-identical parts. In addition, applying modular redundancy can make a system truly fault tolerant. For example, a modular system can shift operation from failed modules to healthy ones, allowing repairs to be carried out without downtime[137]. The design transition from component to modular redundancy actually reduces costs and enhances efficiency, flexibility, and reliability. Despite the various benefits that modularity offers, multilevel modular redundancy allocation optimization has seldom been discussed in detail, nor has an appropriate methodology been provided. To leverage the merits of modular redundancy allocation, this research presents a methodology for optimizing the system reliability of a multilevel class of problems using a modular redundancy allocation scheme.

In a similar direction, Yun and Kim[152] proposed a multilevel series redundancy allocation optimization model in which they considered that each unit of a three level series system is subjected to redundancy, and they optimized system reliability by using conventional genetic algorithms (GAs). Their method can solve certain problems based on the assumption where only one unit is allowed to have redundancy in a direct line. This assumption reduces the feasible design space and fails to yield a globally optimal solution, because conventional GAs require a one-dimensional vector representation of the design variables. Later, Yun et al[161] presented a formulation of multiple multi-level redundancy allocation problems for series systems and applied a GA with a sequential recording method, without reflecting the solution positions. However, the design variables in a multilevel

system have hierarchical relationships, and the artificial transformation into vector coding leads to a reduced feasible design space and suboptimal solutions.

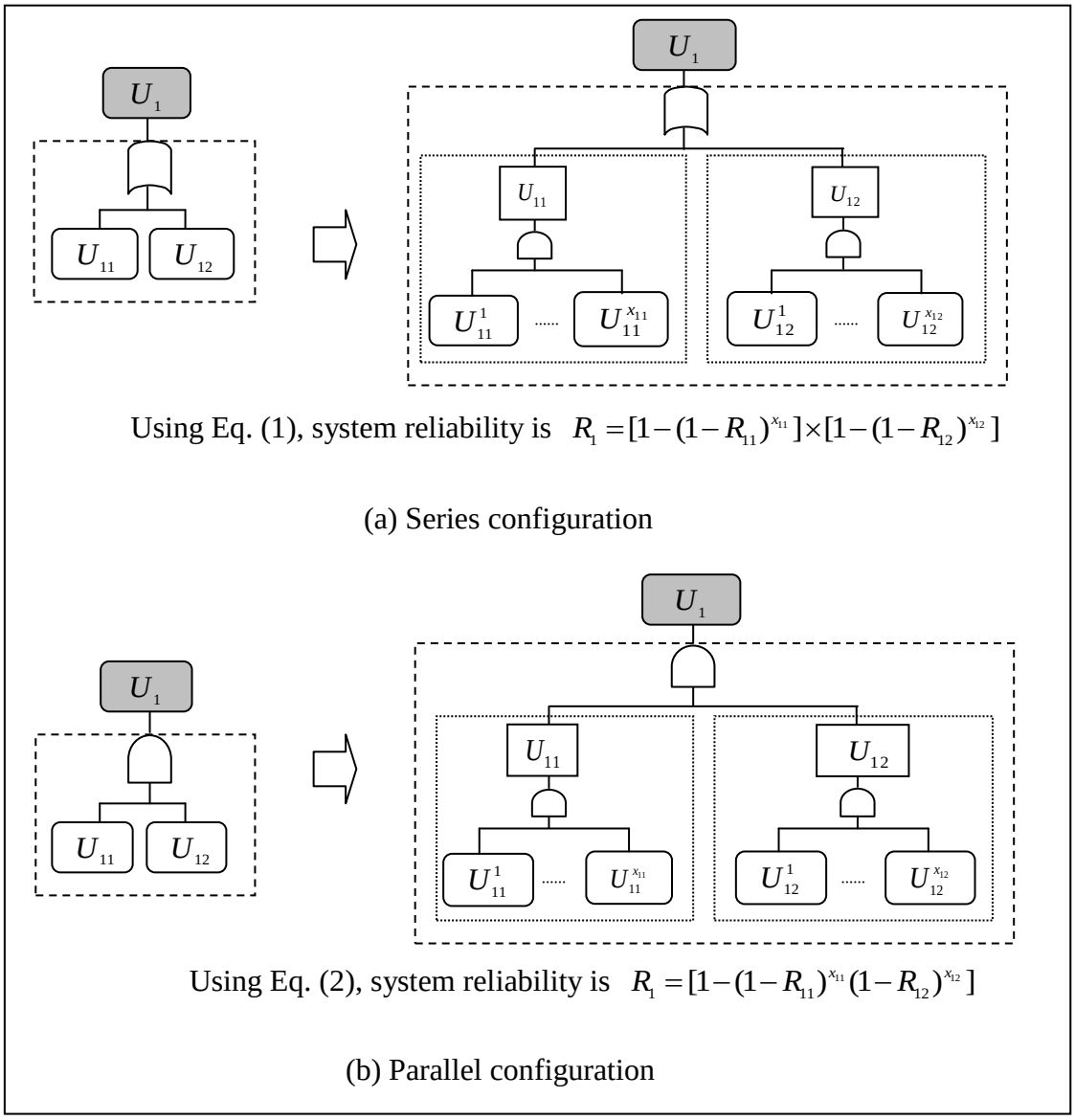
Therefore, this research work proposes a modular redundancy allocation optimization methodology in which hierarchical design variables are represented by hierarchical genotypes in the optimization. This customized methodology is based on a type of genetic algorithm proposed by Yoshimura and Izu[158], in which the hierarchical genotype coding representation is used to exactly express the internal structure and related hierarchical details, a technique using so-called Hierarchical Genetic Algorithms (HGAs). In order to handle general multilevel redundancy allocation problems such as series and series-parallel problems, this research redefines a mathematical expression of system reliability for series and series-parallel and proposes a design-variable coding method using hierarchical genotypes. This research demonstrates that a HGA can handle both modular and component schemes of redundancy allocation easily, by using two newly defined genotypes, nodal and terminal.

This chapter is organized as follows. Section 4.2 describes the detailed mathematical formulation for the multilevel redundancy allocation optimization problems. In Section 4.3, HGA concepts are explained and a HGA coding method for modular redundancy allocation optimization problems is proposed. In section 4.4, we solve two multilevel redundancy optimization problems, one series and one series-parallel, each having four hierarchical levels. In this section, the input data used and the results are summarized. The results obtained in Section 4.4 are explained and discussed in Section 4.5. Finally, Section 4.6 concludes the chapter.

4.2 Modular redundancy allocation in series and series-parallel system

A multilevel redundancy allocation optimization problem is structurally hierarchical, with the system level topmost and the component level at the very bottom. The

subsystems in between the top and the lowest levels are the so-called modules. Each of these modules and their components are termed a unit.



Using Eq. (1), system reliability is $R_1 = [1-(1-R_{11})^{x_{11}}] \times [1-(1-R_{12})^{x_{12}}]$

(a) Series configuration

Using Eq. (2), system reliability is $R_1 = [1-(1-R_{11})^{x_{11}}(1-R_{12})^{x_{12}}]$

(b) Parallel configuration

Fig. 4.2. Series and parallel redundancy allocation in unit U_1.

Fig. 2.6 is a schematic diagram of a general multilevel redundancy allocation configuration. In this figure, U_1 is a system unit containing U_{11} to U_{1,n_1} units as modules at its next lower hierarchical level. Similarly, the U_{11} unit, which is actually the second level of the system hierarchy module, contains n_{11} sub-units as modules or components at its next lower level, represented as U_{111} to $U_{11n_{11}}$. This structure is

replicated until the lowest level of system hierarchy is reached. The connecting lines in the diagram imply the logical relationships among the units at different levels, relationships that may be in series, in parallel, or combinations of these two. Redundancy at all levels is assumed to be active and failures are statistically independent.

The reliability R_i of unit U_i for multilevel series and parallel configurations can be calculated using (2.3), (2.4), and (2.5) given in chapter 2. Fig. 4.2 shows an example of redundancy allocation in unit U_1. Fig. 4.2(a) and Fig. 4.2(b) illustrates the redundancy allocation in a series and parallel system, respectively. The cost constraint of a multilevel redundancy allocation model also reveals hierarchical relationships among the multilevel units. The system cost is essentially the sum of the component and module costs. The assembly costs represent the sum of the costs of adding, duplicating or repairing the module or component. Note that there are definite advantages to using modular redundancy, because the cost of adding, duplicating, or repairing a module is lower than carrying out a similar action upon a component. This is because the lower the level in a system, the more costly the repair job. The expressed cost function will differ depending upon the arrangement of different structures.

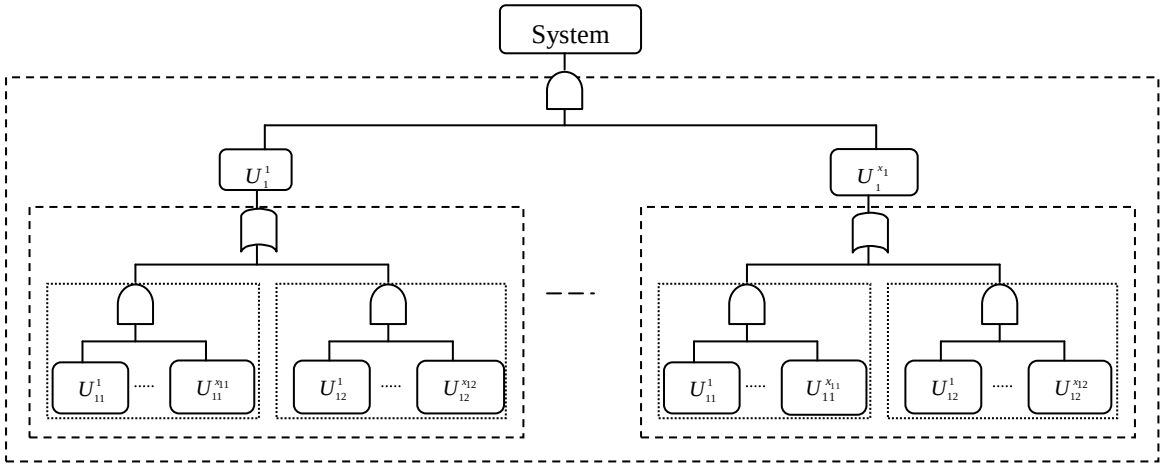


Fig. 4.3. An example of series redundancy allocation in a unit U_1

The redundancy allocation optimization problem in a reliability system consisting of a

set of design variables is expressed as:

$$\text{Maximize} \quad R_s = f(\boldsymbol{x}) \tag{4.1}$$

$$\text{Subject to} \quad C(\boldsymbol{x}) \le C_0 \tag{4.2}$$

where R_s, $f(\boldsymbol{x})$, $C(\boldsymbol{x})$, and $\boldsymbol{x}$ are the system reliability, reliability function, cost function, and a set of design variables, respectively. C_0 is a given fixed positive value for the cost constraint. For example, the problem of optimizing a 2-level series redundancy allocation, as shown in Fig.4.3, can be stated mathematically as follows:

$$R_s = [1-(1-\{1-(1-R_{11})^{x_{11}}\}\times\{1-(1-R_{12})^{x_{12}}\})^{x_1}] \tag{4.3}$$

where, x_1, x_{11}, and x_{12} are the number of redundancy of units U_1, U_{11}, and U_{12}, respectively. The values of the design variables, x_{11}, and x_{12} depend on the value of x_1, the design variable of the parent unit. If the number of redundancies represented by x_1 is two, then the redundancies of x_{11}, and x_{12} should be at least two, however the values of design variables x_{11}, and x_{12} are independent of each other.

In this chapter, the following cost functions have been applied to calculate the costs for modules and components:

$$C_i = \sum_{m=1}^{n_i}\sum_{j=1}^{x_i} C_{i,m}^{j} \times x_i \tag{4.4}$$

$$C_i = c_i x_i + \lambda_i^{x_i} \tag{4.5}$$

where $C_{i,m}^{j}$ are the modular costs of sub-units $U_{i,m}^{j}$. The symbols x_i, c_i, and λ_i respectively represent the redundancy number, the unit cost, and the assembly cost for the *i*-th unit. Each $C_{i,m}^{j}$ value is calculated using Eq (4.4) at the level immediately below the unit, and these calculations are recursively iterated to the level just above the very lowest hierarchical level. At the very lowest level, where there are no sub-units belonging to unit U_i, the cost is calculated using Eq (4.5). Eq (4.5) first appears in the paper of Yun and

Kim[152]. Thus, the total cost for a multilevel structure is calculated by using Eq(4.4) and Eq(4.5).

4.3 Hierarchical Genetic Algorithm for series and series-parallel problems

Hierarchical Genetic Algorithms[158] are customized and applied to solve the multilevel redundancy allocation optimization problems here. HGAs are advanced genetic algorithms that can represent hierarchical relationships among design variables using hierarchical genotypes, and can optimize hierarchical problems in a single optimization process. While conventional genetic algorithms[122] use vector genotype structures, HGAs employ hierarchical genotype structures.

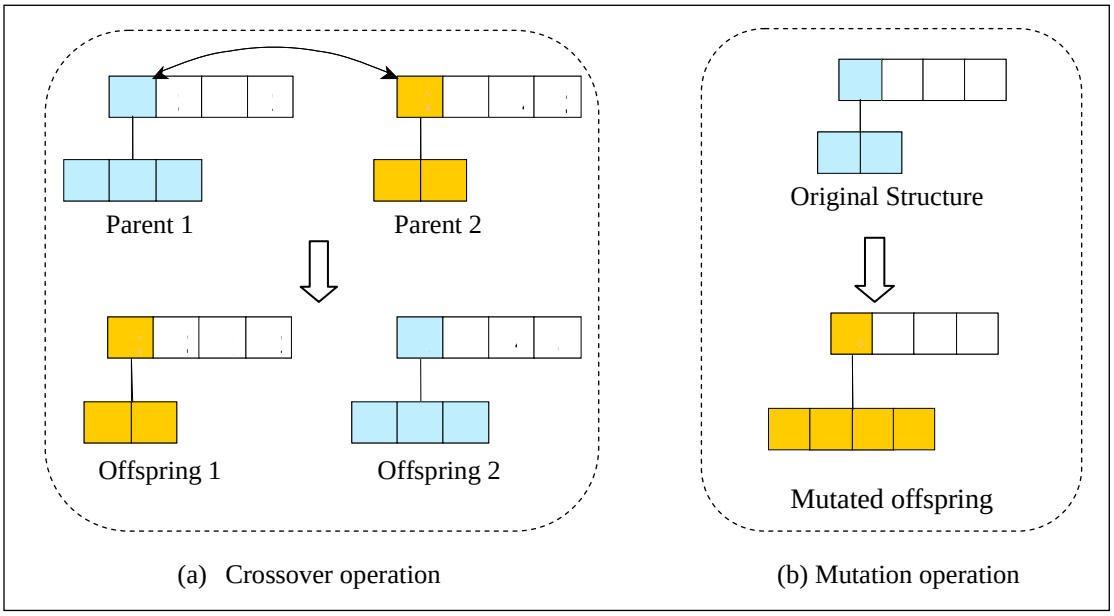


Fig. 4.4. Crossover and mutation operators for hierarchical genotype.

The multilevel redundancy allocation optimization problems here involve hierarchical relationships among design variables, which represent redundant modules or component selections and hierarchical genotype representation is particularly suited to handling such hierarchical relationships. Since HGAs have special types of genotype structures, new crossover and mutation operators have to be applied. The HGAs allow branches of the hierarchical structure to be exchanged, in addition to the exchange of genes. Fig.4.4

illustrates the crossover and mutation operators for hierarchical genotypes. Using such genetic operations, new individuals are produced and optimal hierarchical structures can then be obtained.

4.3.1 Solution encoding

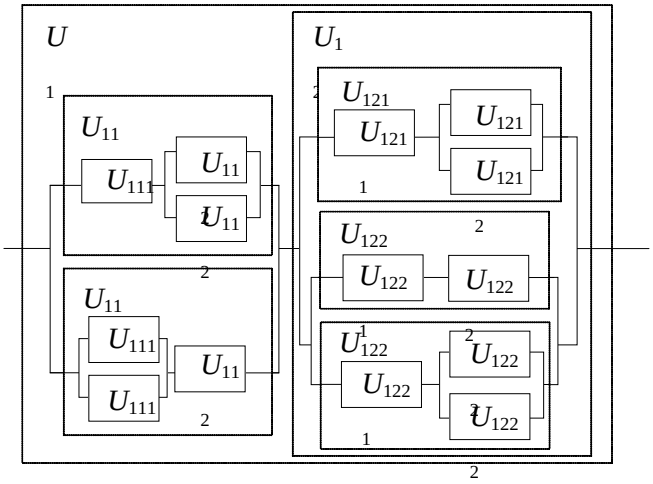
A hierarchical genotype is represented here using two types of node, ordinal and terminal, as shown in Table 4.1. Ordinal node N_i corresponds to redundancy unit U_i, and is characterized by several parameters and design variables. Parameter $T \in \{\mathrm{S},\mathrm{P}\}$ represents the type of unit where S means that the sub-units have a series reliability relationship, while P means a parallel configuration. When *T*=S, this node is called a series node, and when *T*=P, the node is called a parallel node. Parameters *k* and *n* stand for the redundancy number of unit U_i and the number of sub-units, respectively. Here, *k* is given by a design variable at an upper node, while the parameter *n* is a fixed value that depends on the optimization problem to be solved. $x_{i,m}^{j}$ is a design variable denoting the redundancy number for the *m*-th sub-unit of the *j*-th redundancy unit, where *j* varies from 1 to *k*. Therefore, there are $n_i k$ design variables in unit U_i. A terminal node N_{t_i} corresponds to one of the lowest units, and incorporates design variable *k*, unit reliability r_i, and the unit cost c_i. Since there are no sub-units at the terminal node, it does not contain parameter *n* or design variable $x_{i,m}^{j}$. Using these two genotypes, all possible redundancy allocation solutions for both the series and series-parallel reliability allocation problems can be represented.

TABLE 4.1

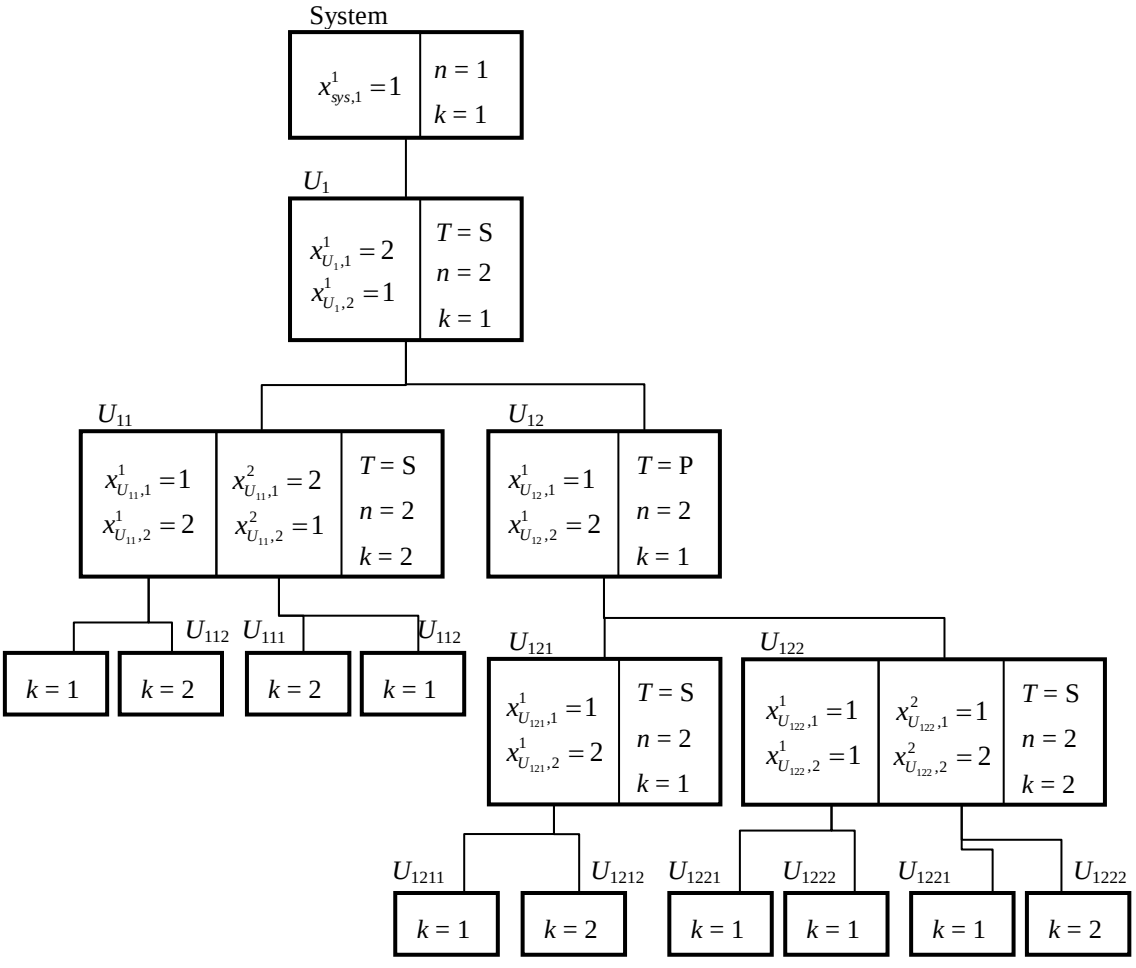
HIERARCHICAL GENOTYPE REPRESENTATION FOR SERIES AND SERIES-PARALLEL SYSTEM

	Ordinal genotype node N_i	Terminal genotype node N_{t_i}
Design variable	$x_{i,m}^{j}$: the number of subordinate modules for the *m*-th module	
Parameter	*T*: unit type *n*: the number of sub-modules *k*: the redundancy for unit U_i	*k*: the redundancy for component U_i r_i : unit reliability c_i : unit cost

Fig. 4.5 illustrates an example of the genotype encoding. Fig. 4.5(a) shows a redundancy configuration for a system U_1 consisting two modules, U_{11} and U_{12}, at the second level. This redundancy structure can be represented using hierarchical genotype nodes as shown in Fig. 4.5(b). The ordinal and terminal nodes are assigned to represent module and component units at each level. Note that unit features, such as the number of redundant units and series or parallel configuration, are expressed in the corresponding upper unit node. This redundancy allocation solution has two redundant units for U_{11}, and this feature is characterized using $x_{U_1,1}^{1} = 2$ in the U_1 node. Furthermore, the parallel relationship between U_{121} and U_{122} is described as T = P in node U_{12}. Thus, a single system node exists in this representation scheme in order to denote that U_1 has only a single redundancy unit. Note that the units that have series and parallel relationships cannot share the same upper unit.



(a) An example of a multilevel reliability system U_1



(b) Design variables and parameters at each ordinal and terminal node

Fig. 4.5. Hierarchical genotype representation in system U_1.

When there is a mixture of series and parallel configurations at the same level, as

shown in Fig. 4.6, U_1, U_2, U_3, and U_4 cannot be directly encoded into the hierarchical genotype. In this case, a new unit, $U_{\square}$, which represents the grouping of U_2 and U_3, is introduced and the reliability system is encoded using a series node, i.e., T = S, to represent that its sub-units are U_1, $U_{\square}$ & U_4, and U_2 & U_3 are then encoded as sub-units of $U_{\square}$.

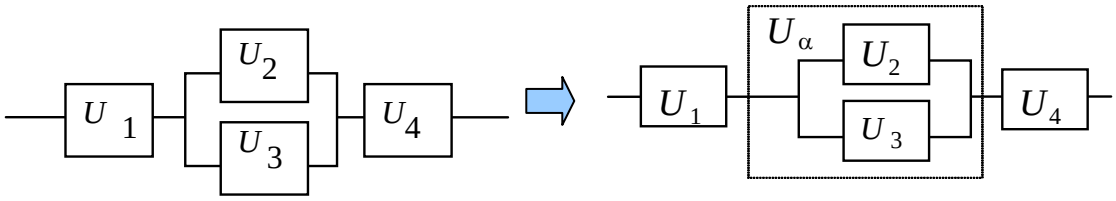


Fig. 4.6. Interpretation of mixed series and parallel configurations.

The HGA example shown in Fig. 4.5(b) illustrates that genotypes using fixed arrays, frequently used in various optimization problems, are not applicable to this problem since the number of design variables varies according to the number of redundant units. Here, the two design variables, $x^1_{U_{11},1}$ and $x^2_{U_{11},1}$, represent the redundancy of U_{111}, since there are two redundant units for U_{11}, which is the unit above U_{111} in the hierarchy. If the number of redundant units for U_{11} increases, the number of design variables for U_{111} will also increase. The solution encoding scheme proposed in this research can successfully represent different numbers of design variables at every hierarchical level.

The ordinal and the terminal genotypes each have two functions, namely, reliability and cost, and the difference between series and parallel nodes only pertains to reliability calculations. When the reliability function in the series node is called, the unit reliability is calculated using Eq. (2.3), while Eq. (2.4) is used for the parallel node. When calculating either of these equations, the reliability values of the lower units, $R^k_{i,m}$, are required, and these are obtained by calling the reliability function of the lower units. Finally, the reliability function of the terminal node returns its unit reliability, r_i. Thus, the reliability functions are recursively called and the total system reliability can be effectively obtained.

Similarly, the system cost can be obtained by calling the cost function embedded in each node.

4.4 Numerical Examples

4.4.1 Four level series and series-parallel problems

In this section, we solve two multilevel redundancy allocation optimization problems. Fig.4.7 and Fig.4.8 show the two 4-level multilevel systems that have series and series-parallel configurations. We applied the HGA to optimize the system reliability of these two problems. For example, U_1 is a unit at the system level, (U_{11}, U_{12}) and (U_{111}, U_{112} U_{121},U_{122}) are units at module levels, and (U_{1111}& U_{1112},U_{1121}& U_{1122}, U_{1211}&U_{1212}, U_{1221}&U_{1222}) are units at the component level.

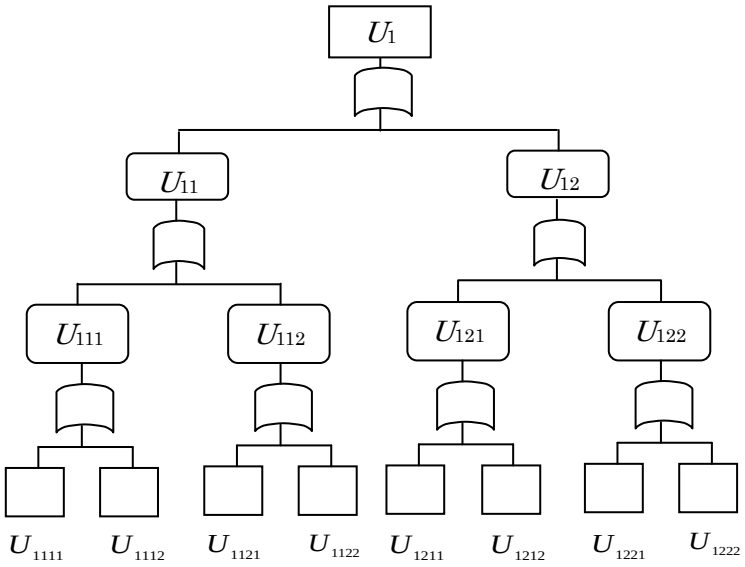


Fig. 4.7. Four-level hierarchical series configuration of U_1.

Fig.4.7 shows a series system in which all the units are arranged in series at every level, while Fig.4.8 shows a series-parallel system in which U_{121} & U_{122} and U_{1121} & U_{1122} are in parallel and the rest of units are in series either at the same level or at different levels. We see in Fig.4.8 that U_{12} & U_{112} consists of parallel units, U_{121} & U_{122} and U_{1121} & U_{1122} at their immediate lower levels.

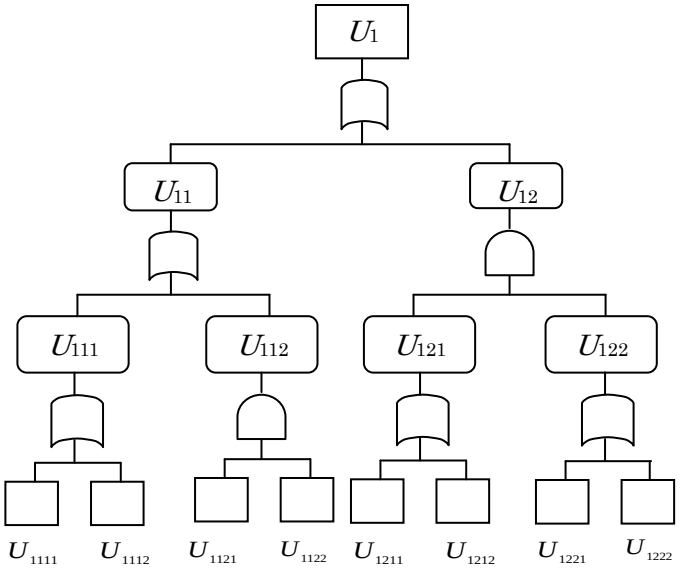


Fig. 4.8. Four-level hierarchical series-parallel configuration of U_1.

4.4.2 Input data

Suitable parameters for optimizing the two allocation problems were selected based on several experimental runs using the proposed HGA. The crossover rates, p_{c_1} and p_{c_2}, when solving these problems, were respectively set to 0.8 & 0.5 and the mutation rate p_m was set to 0.05. An initial population of 100 individuals was generated, and 500 generations were processed in each case. Table 4.2 summarizes the basic reliability and corresponding cost for each unit in both problems. The unit reliability and the unit cost at the very lowest level in the multilevel redundancy allocation problems were used when calculating the unit reliability and the unit cost of upper level units, up to the system level. In each of these tables, x's represent the integer value of the optimal redundancy to be obtained during the optimization process.

TABLE 4.2

INPUT DATA

Level	Unit	Parent unit	Redundancy	Basic Reliability		Cost	λ
				Series system	Series-parallel		
1	U_1		x_1	0.2198	0.6268	102	2
2	U_{11}	U_1	x_2	0.5130	0.7110	48	2
	U_{12}	U_1	x_3	0.4284	0.8816	50	2
3	U_{111}	U_{11}	x_4	0.7200	0.7200	21	3
	U_{112}	U_{11}	x_5	0.7125	0.9875	21	3
	U_{121}	U_{12}	x_6	0.6300	0.6300	23	3
	U_{122}	U_{12}	x_7	0.6800	0.6800	21	3
4	U_{1111}	U_{111}	x_8	0.9000	0.9000	7	4
	U_{1112}	U_{111}	x_9	0.8000	0.8000	6	4
	U_{1121}	U_{112}	x_{10}	0.7500	0.7500	8	4
	U_{1122}	U_{112}	x_{11}	0.9500	0.9500	5	4
	U_{1211}	U_{121}	x_{12}	0.7000	0.7000	9	4
	U_{1212}	U_{121}	x_{13}	0.9000	0.9000	6	4
	U_{1221}	U_{122}	x_{14}	0.8500	0.8500	5	4
	U_{1222}	U_{122}	x_{15}	0.8000	0.8000	8	4

4.4.3 Computational results

The HGA was applied to solve the series and series-parallel problems using separate modular and component redundancy schemes, under the same HGA parameters. In the modular redundancy scheme, we allowed potential redundancy for units at all levels, whereas in the component scheme, we only allowed redundancy at the component level. We applied these two schemes to explore what the differences in the optimal solutions would be under the same cost constraint. Ten cases that used varying unit reliability values in basic configurations were considered when solving the two problems, and ten 500-generation trials were performed in each case.

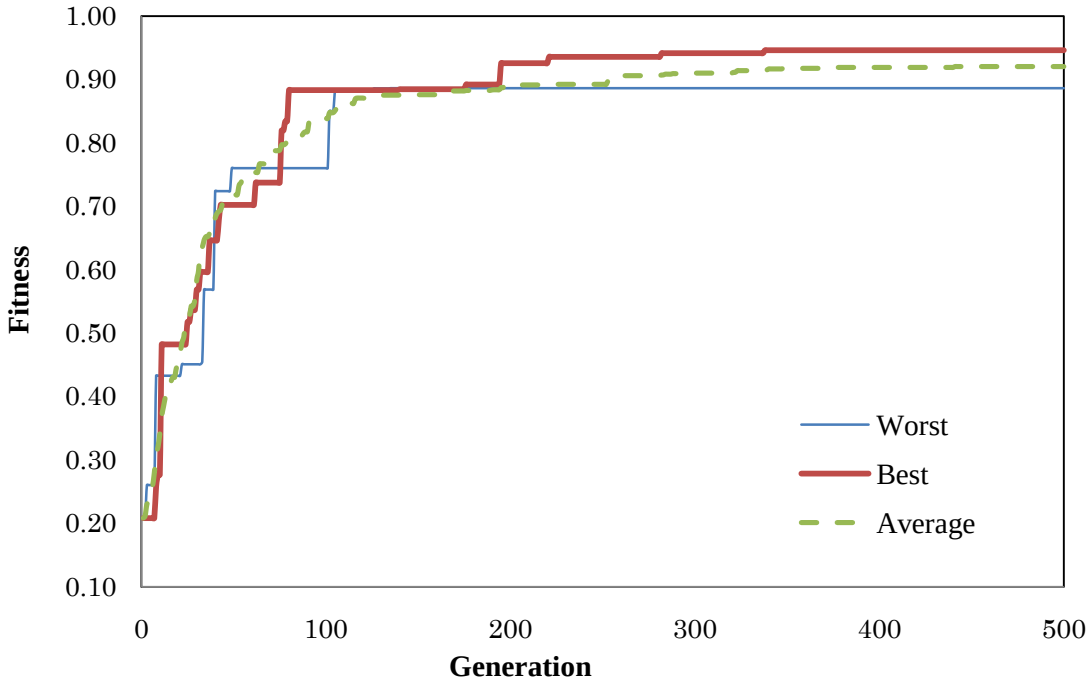


Figure 4.9. Convergence of fitness value in hierarchical series configuration.

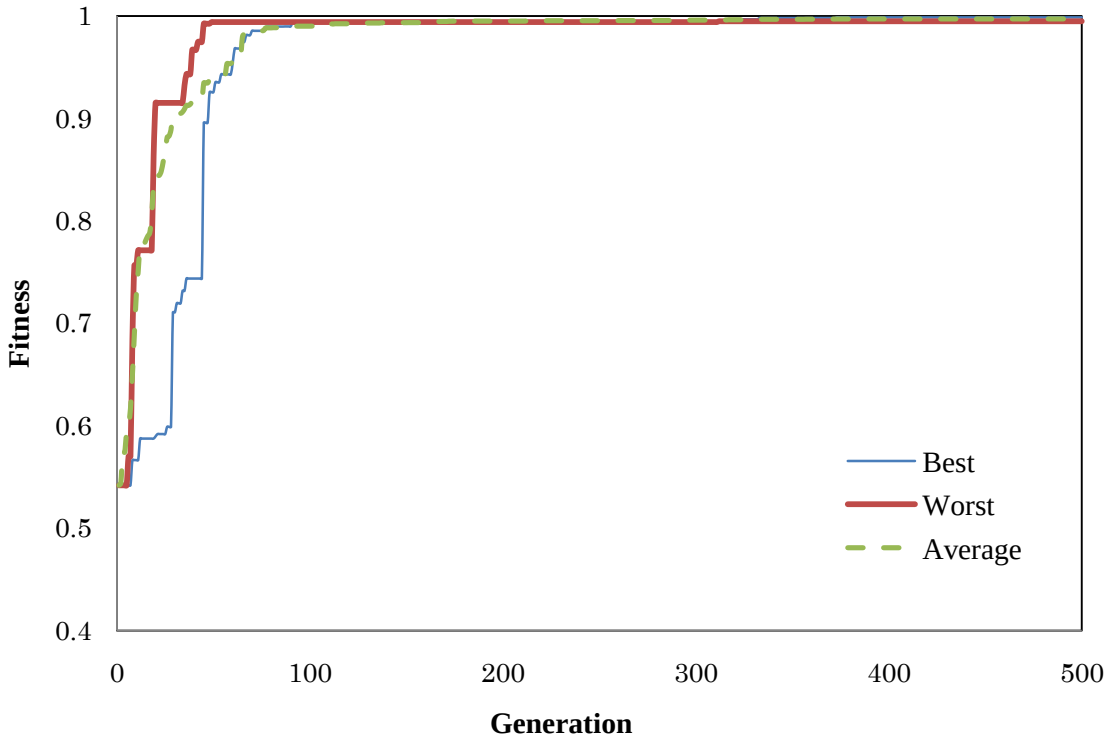


Figure 4.10. Convergence of fitness value in hierarchical series-parallel configuration.

TABLE 4.3

OPTIMAL REDUNDANCY ALLOCATION IN SERIES SYSTEM

Basic Reliability	Modular Redundancy		Component Redundancy	
	Optimal Configuration x_1 - x_2x_3 - x_4x_5 - x_8x_9 - $x_{10}x_{11}$ - x_6x_7 - $x_{12}x_{13}$ - $x_{14}x_{15}$	Optimal Reliability	Optimal Configuration x_1 - x_2x_3 - x_4x_5 - x_8x_9 - $x_{10}x_{11}$ - x_6x_7 - $x_{12}x_{13}$ - $x_{14}x_{15}$	Optimal Reliability
0.3202	1-22-11-1222-2222-21-32-33	0.9775	1-11-11-23-32-11-32-23	0.9460
0.1050	1-11-21-2212-33-12-23-2222	0.8677	1-11-11-33-23-11-22-32	0.7658
0.3202	1-11-22-2212-2122-12-33-2222	0.9777	1-11-11-32-23-11-23-32	0.9460
0.1201	1-11-12-32-2223-12-22-2222	0.8870	1-11-11-22-33-11-22-33	0.7857
0.1587	1-11-12-32-2223-12-22-2222	0.9368	1-11-11-33-32-11-32-22	0.8427
0.1805	1-11-32-112222-2123-21-2211-32	0.9508	1-11-11-23-23-11-23-32	0.8578
0.2318	1-11-12-23-1212-22-2132-2222	0.9538	1-11-11-23-22-11-32-23	0.9094
0.2938	1-11-12-23-2122-22-2232-32212	0.9742	1-11-11-32-32-11-23-32	0.9376
0.2179	1-11-22-2222-2222-21-2222-32	0.9502	1-11-11-32-32-11-23-32	0.8945
0.2085	1-11-23-2122-112112-22-2222-2132	0.9645	1-11-11-33-32-11-22-32	0.8928

Fig.4.9 and Fig.4.10 show the convergence of objective function values in ten 500-generation for both the problems. The cost constraint was always kept constant at a value of 500. Finally, the best solution among the 10 trials is summarized for the two problems in Table 4.3 and Table 4.4. Optimal redundancy allocations and solutions are

given in both tables. Table 4.3 provides the best solutions for the series redundancy allocation problems and Table 4.4 the best solutions for the series-parallel problem, and the reliability settings are the same in each row of the tables. Here, reliability settings refer to the number of units, the hierarchical levels, and the reliability of the units.

TABLE 4.4

OPTIMAL REDUNDANCY ALLOCATION IN SERIES-PARALLEL SYSTEM

Basic Reliability	Modular Redundancy		Component Redundancy	
	Optimal Configuration x_1-x_2x_3-x_4x_5-x_8x_9-$x_{10}x_{11}$-x_6x_7-$x_{12}x_{13}$-$x_{14}x_{15}$	Optimal Reliability	Optimal Configuration x_1-x_2x_3-x_4x_5-x_8x_9-$x_{10}x_{11}$-x_6x_7-$x_{12}x_{13}$-$x_{14}x_{15}$	Optimal Reliability
0.3202	1-11-21-2323-22-11-32-22	0.9991	1-11-11-23-32-11-32-23	0.9422
0.1050	1-11-31-221223-22-31-222221-22	0.9956	1-11-11-33-23-11-22-32	0.7658
0.3202	1-11-22-2332-2211-21-2222-22	0.9993	1-11-11-32-23-11-23-32	0.9460
0.1020	1-11-22-3232-2211-21-2122-22	0.9986	1-11-11-31-33-11-22-33	0.7857
0.1587	1-11-31-222322-22-21-1122-23	0.9967	1-11-11-33-22-11-32-22	0.8427
0.1805	1-11-41-22232212-23-11-22-22	0.9975	1-11-11-23-23-11-23-32	0.8578
0.2318	1-11-31-222221-32-12-21-2322	0.9995	1-11-11-23-23-11-32-23	0.9094
0.2938	1-11-31-221322-21-31-112223-12	0.9995	1-11-11-23-22-11-33-23	0.9377
0.2179	1-11-21-3232-22-21-2312-22	0.9980	1-11-11-32-32-11-23-32	0.8945
0.2085	1-11-31-222222-13-11-33-22	0.9986	1-11-11-33-32-11-22-32	0.8928

Table 4.3 and Table 4.4 indicate that the optimal component redundancy allocation is the same in each case, implying that the achieved values represent optimal system reliability. However, the optimal modular redundancy allocation and the corresponding system reliability for the series and series-parallel systems are different. For example, in the tenth case of Table 4.4, the optimal modular redundancy configuration for U_1 at the system level, [U_1], is [1], at the second level, [$U_{11}U_{12}$], is [11], at the third level, [$U_{111}U_{112}$-$U_{121}U_{122}$], is [31-11], and at the lowest level, [$U_{1111}U_{1112}$-$U_{1211}U_{1212}$-$U_{1211}U_{1212}$-$U_{1221}U_{1222}$], is [222222-13-33-22]. Fig.4.11 shows an optimal redundancy arrangement in the modular schemes.

For the same case in Table 4.4, the optimal component redundancy configuration for U_1 at the system level, [U_1], is [1], at the second level, [$U_{11}U_{12}$], is [11], at the third level, [$U_{111}U_{112}$-$U_{121}U_{122}$], is [11-11], and at the lowest level, [$U_{1111}U_{1112}$-$U_{1211}U_{1212}$-$U_{1211}U_{1212}$-$U_{1221}U_{1222}$], is [33-32-22-32]. Fig. 4.12 shows an optimal redundancy arrangement for the component scheme. All the optimal solutions summarized in Table 4.3 and Table 4.4 can be illustrated by pictorial representations in a similar way.

Next, we solved the two allocation optimization problems by varying the cost constraints. Ten cases using various cost constraints while maintaining constant values of unit reliability were considered when solving the series and series-parallel problems, 10 500-generation trials were performed in each case. Finally, the best solution among the 10 trials was selected as the optimal solution. Fig. 4.13 and Fig. 4.14 graphically show the trends of the optimal solutions when plotted against the cost constraints for both systems. In both problems, the cost constraint was varied in increments of 50, from 200 to 650.

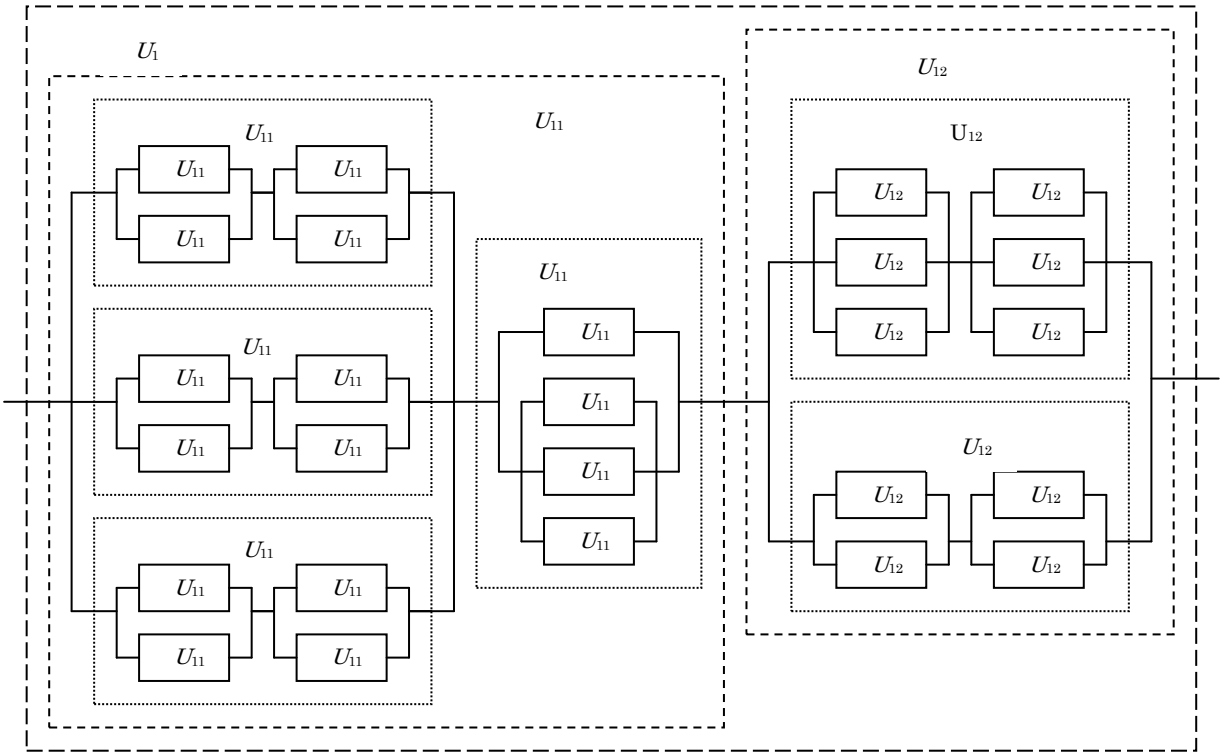


Fig. 4.11. Optimal modular allocation in 4-level series-parallel system.

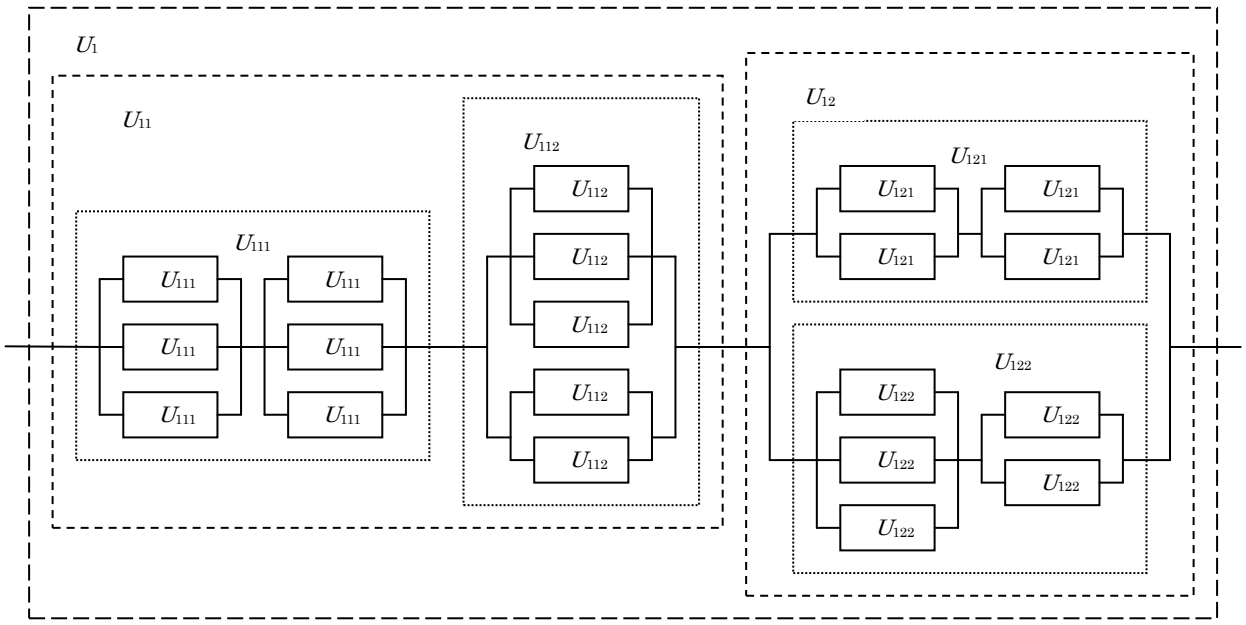


Fig. 4.12.Optimal component allocation in 4-level series-parallel system.

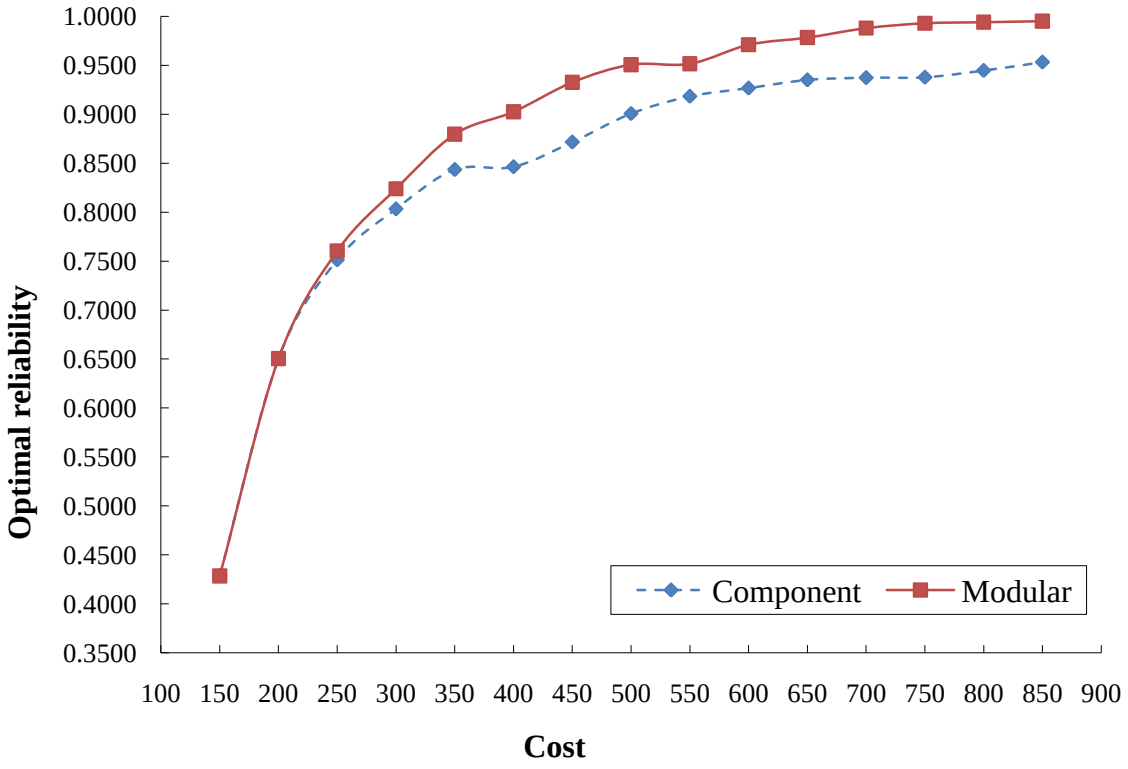


Fig. 4.13. Modular and component redundancy allocations in series system.

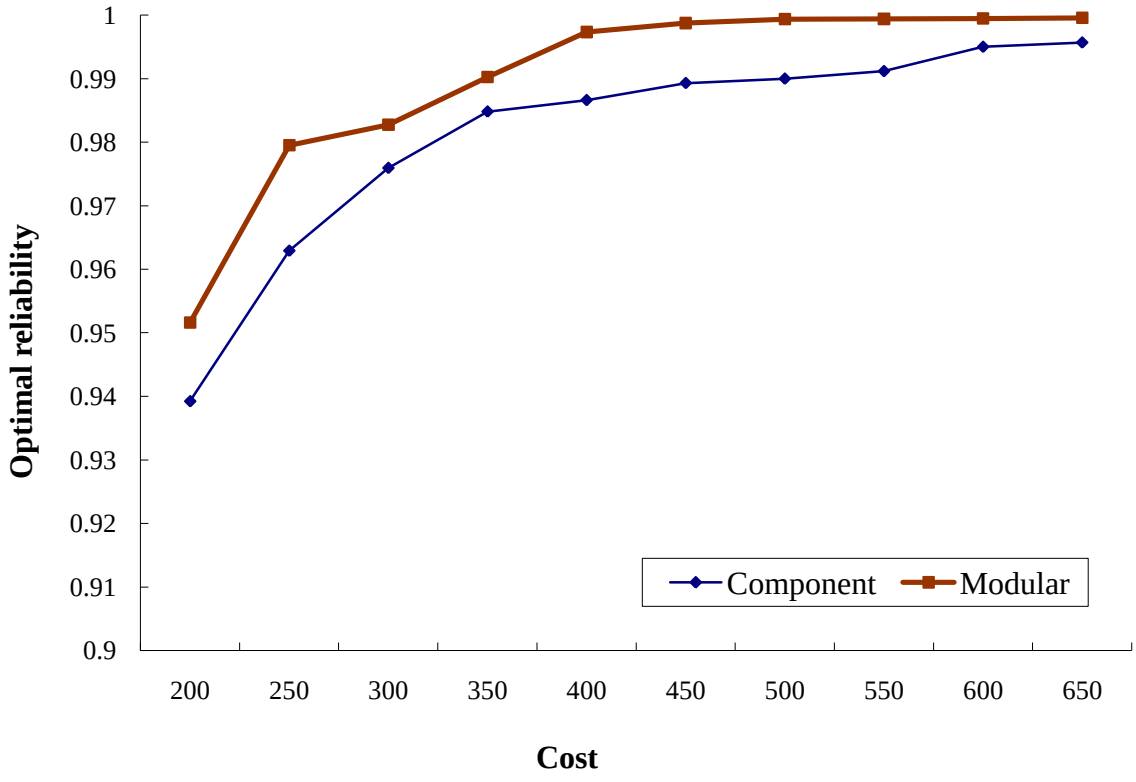


Fig. 4.14. Modular and component redundancy allocations in series-parallel system.

In both figures, we compare the optimal solutions obtained using a modular redundancy scheme with those obtained using the component redundancy scheme. The *x*- and *y*-axes respectively represent the system cost constraints and the optimal system reliability.

4.5 Discussion

The numerical examples for the multilevel series and series-parallel redundancy allocation problems clearly demonstrate that the modular scheme of redundancy allocation has certain distinct advantages over the component scheme of redundancy allocation. The obtained results shown in Table 4.3 and Table 4.4 clearly support the claim that a modular redundancy approach yields superior system reliability compared with the component redundancy scheme. We see in Table 4.4 that the average improvement in optimal solutions using a modular redundancy scheme for a multilevel series system is 7.6%, and the maximum improvement is 13.3% better than the best result obtained with the component scheme of redundancy allocation. Similarly, for the series-parallel system, the modular redundancy approach yielded an average improvement of 13.8% and a maximum improvement of 30.0% compared with the component scheme of redundancy allocation. Although the percentage improvement can vary according to the parameters used, we infer that modular redundancy yields better optimal solutions than component redundancy scheme in multilevel redundancy allocation optimization problems.

The average computation time in single run for modular allocation optimization problems varies between 149.0 to 153.1 seconds. On the other hand, the average calculation time in component allocation problems lie between 19.6 and 20.2. Thus, the computational effort in solving modular redundancy allocation optimization problems is more because of larger search space size than component allocation problems. Furthermore, Fig.4.12 and Fig.4.13 indicate that the optimal reliability achieved for each of the ten cost

constraint cases again demonstrates the superiority of the modular redundancy scheme over the component scheme. Although, the difference in the optimal solutions between the two schemes is not very significant at a system cost of 200, higher system cost values increasingly show the superiority of a modular allocation approach.

Thus, it appears advantageous to allocate redundancy without affecting the internal hierarchical relationships of a multilevel reliability system. It is recognized that using conventional GAs to represent design variables having hierarchical relationships is problematic, and we overcame this difficulty by applying HGAs in which the modular design variables are encoded using an innovative hierarchical genotype representation. We observe that the hierarchical genotype representation of modular design variables is highly appropriate for solving hierarchical reliability optimization problems, since such representation allows sufficient flexibility for every possible redundancy combination to be addressed.

As described in the introduction, the particular benefit of using a modular approach to redundancy allocation in a multilevel system, and hierarchical genotypes, is that both fault tolerance and system reliability are improved. Modularity reduces the number of parts and thus simplifies the system design. The fewer parts and subsystems there are, the more reliable a system will be in service. Thus, well-implemented modular redundancy offers a kind of synergistic benefit in terms of reducing complexity while increasing the fault tolerance of a system design, and the computational results presented confirm that modular redundancy allocation optimizations lead to improved optimal system reliability. This is because a hierarchical genotype representation not only preserves the hierarchical relationship of the modular design variables but also allows simultaneous redundancy allocation at more than one level during the optimization process.

4.6 Summary

This chapter discussed the importance of modular redundancy allocation applied to multilevel system reliability problems. We proposed a methodology to solve series and series-parallel redundancy allocation problems considering the hierarchical relationships among design variables. Modular design variables were encoded using hierarchical genotypes in hierarchical genetic algorithms, and the multilevel redundancy allocation optimization problems were efficiently solved. The optimization of numerical examples in this chapter indicates that the modular scheme of redundancy allocation yields superior system reliability for multilevel configurations, in contrast to the conventional notion that component level redundancy allocation yields better optimal solutions. The application of a HGA proved to be flexible and efficient when solving large-scale multilevel redundancy allocation optimization problems.

Chapter 5

Multiobjective hierarchical genetic algorithms for optimal reliability design

5.1 Introduction

Most complex engineering systems exhibit hierarchical design structures and the reliability of such systems during the design stage can be optimized either by enhancing component reliability or by allocating appropriate redundancy at the component level. The latter technique is widely practiced in industry when designing systems that must be highly reliable. Problems whose solutions aim to provide optimum redundancy to the units located at multiple levels of a complex system, subject to certain resource constraints, are generally termed multilevel redundancy allocation optimization problems (MRAOPs).

Apart from being NP hard[2], MRAOPs involve hierarchical design variables that require certain basic structural relationships to be maintained throughout the optimization process. MRAOPs therefore require a suitable algorithm that allows appropriate representation of hierarchical design variables during the optimization process. Moreover, the size of the search space when optimizing MRAOPs tends to be very large, so that using exact methods to solve such problems may too computationally costly. Techniques using GAs[122] are therefore attractive when solving difficult optimization problems and an additional reason for their popularity is that they can be customized to solve a particular problem by introducing user-defined encoding schemes, selection strategies, crossover operators, and mutation operators. Being stochastic, GAs do not guarantee true globally optimal solutions, but solutions that approach globally optimum solutions can be easily found.

Recently, the growing research interest in multilevel reliability modeling and

optimization using GAs is reflected in the literature due to the practical importance of these techniques. Yun and Kim[152] have solved MRAOPs in series systems having three hierarchical levels using a customized GA. Later, Yun et al.[161] presented a formulation for multiple multi-level redundancy allocation problems for series systems and applied a GA with a sequential recording method, without reflecting the solution positions. Levitin[140] proposed an algorithm for solving multilevel protection cost minimization problems subject to survivability constraints. This algorithm is based on a universal generating function technique used for system survivability evaluation and on a genetic algorithm used as an optimization engine. However, almost all of the research has been limited to single objective (SO) optimization problems for maximizing system reliability subject to a cost constraint.

When solving MRAOPs using GA techniques, two important issues need to be addressed: how best to represent hierarchical design variables, and how to search for the best solution most efficiently despite a very large solution space, particularly for problems having more than three hierarchical levels. The first issue can be resolved by using a HGA, such as proposed by Kumar et al.[162], that uses a hierarchical genotype encoding scheme for the MRAOP design variables. However, the second issue, that of searching efficiency for MRAOPs, has yet to be addressed for single objective HGAs because they use an elite-preservation strategy which fails to preserve adequate population diversity, so the best solutions are often overlooked[163]. Therefore, the need to apply a diversity preservation mechanism in selection operators to enhance the yield of optimal solutions during optimization process is clear, and one way of preserving such diversity is to introduce a multiobjective (MO) scheme for solving MRAOPs.

In a practical sense, MO optimization is preferable because it provides a decision-maker with several trade-off solutions to choose from. Furthermore, practical

engineering reliability problems actually do have multiple conflicting objectives such as maximization of reliability and performance while minimizing cost and weight, and so on. Multiple objective formulations are practically required for concurrent optimization that yields optimal solutions that balance the conflicting relationships among the objectives. MO optimization yields a set of Pareto-optimal solutions, which is a set of solutions that are mutually nondominated[164]. The concept of nondominated solutions is required when comparing solutions in a multidimensional feasible design space formed by multiple objectives. When two conflicting objectives are present, such as when seeking to maximize reliability while minimizing cost, there will always be a certain amount of sacrifice in one objective to achieve a certain amount of gain in the other when moving from one Pareto solution to another. But decision-makers often prefer to use a Pareto optimal solution set rather than being provided with a single solution, because the set helps them effectively understand the trade-off relationships among conflicting objectives and make informed selections of the best solutions to practical engineering problems.

It is important to note that MO optimization of MRAOPs is more difficult than SO optimization because the former handles two goals, progressing towards the Pareto-optimal front and maintaining a diverse set of solutions in the nondominated front, while the latter has only a single goal of searching for an optimal solution. Moreover, exact methods are very time consuming in MO optimization since the objective function space in MO is multidimensional unlike the single objective function space in SO optimization problems. MO optimization difficulties can be alleviated by avoiding multiple simulation runs, doing without artificial aids such as weighted sum approaches, using efficient population-based evolutionary algorithms, and the concept of dominance[163]. In addition, the use of multiobjective genetic algorithms (MOGAs) provides a decision-maker with the practical means to handle MO optimization problems.

Given the above concerns, this chapter aims to address two particular issues when solving MRAOPs: the suitable representation of hierarchical design variables and the preservation of population diversity in the selection strategy. To achieve these goals, we propose a MO formulation for multilevel series redundancy allocation problems and a methodology to solve them. In this methodology, a general framework of multiobjective hierarchical genetic algorithms is developed by integrating two different approaches, namely, a hierarchical genotype representation for the design variables, and a user defined selection operator with diversity preservation mechanism. In this chapter, we implemented the non-dominated sorting genetic algorithm (NSGA-II)[164] and the strength Pareto evolutionary algorithm (SPEA2)[165] in the selection operators, both of which include an excellent mechanism for preserving population diversity. Additionally, the hierarchical genotype coding scheme is modified to accommodate MRAOPs design variables that have serial linkages and a modular structure. The proposed approach is applied in solving two hierarchical series system MRAOPs, one with three levels and the other with four. We also conduct a SO optimization using a HGA so that the best solution obtained using this method can be compared with those of nearest best solutions on the Pareto-optimal fronts obtained using the NSGA-II and SPEA2.

The rest of the chapter is organized as follows. Section 2 describes a multilevel redundancy allocation problem and its MO optimization formulation. Section 3 provides the details of the MOGAs and the proposed framework of the multiobjective HGA approach. In Section 4, the two numerical examples are solved and computational results are presented. Discussion of the obtained results is presented in Section 5 and Section 6 presents our conclusions.

5.2 Multiobjective formulation of multilevel redundancy allocation optimization problems

The hierarchical structure of a reliability system is shown in Fig.5.1 in which the system level is the topmost level and the component level is the lowest. Subsystem or module levels are located between the top level and the bottom level. Each system, module and component is here termed a unit. System and module units can have any number of subordinate units, such as modules that make up the system or components that make up a module. These subordinate units are called sub-units, and the next highest hierarchical unit of a sub-unit is called a parent unit. The proposed redundancy allocation model can handle redundancy for all units of a multilevel reliability system. The multilevel reliability allocation formulation presented here allows the units to have redundancy not only at the same level, but also simultaneously for sub-units at lower levels.

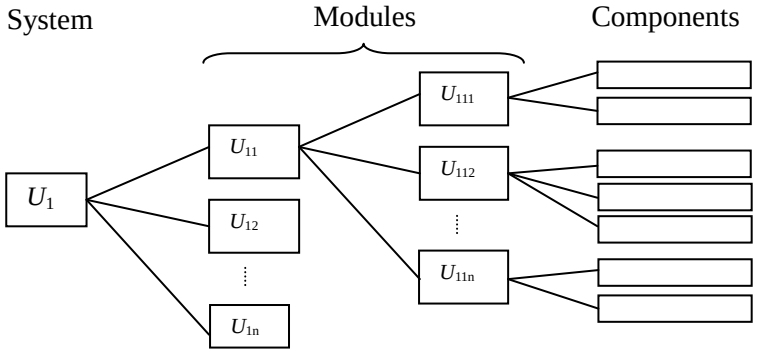


Fig.5.1. Multilevel configuration of system reliability.

As described in chapter 2, the reliability of a unit with multilevel series configurations can be calculated using the following equations:

$$R_i = \prod_{m}^{n_i} [1 - \prod_{j=1}^{x_i} (1 - R_{i,m}^{j})] \tag{5.1}$$

$$R_i = 1 - \prod_{j=1}^{x_i} (1 - R_i^{j}) \tag{5.2}$$

The system cost is usually calculated as the sum of the cost of subsystems and modules, and the cost of a module is the sum of all modules or component costs therein, when there are parallel units in the level immediate below. In practical systems, it is assumed that

multilevel redundancy incurs additional costs, due to the adding or duplication of redundant units to modules, and the increased number of components. In general, the redundancy cost of can be expressed mathematically as follows.

$$C_i = \sum_{m=1}^{n_i} \sum_{j=1}^{x_i} C_{i,m}^{j} \times x_i + \text{additional costs} \tag{5.3}$$

5.2.1 Single objective redundancy allocation optimization formulation

The single objective redundancy allocation optimization problem in a reliability system consisting of a set of design variables is expressed as:

$$\text{Maximize} \quad R_s = f(\boldsymbol{x}) \tag{5.4}$$

$$\text{Subject to} \quad C(\boldsymbol{x}) \le C_0 \tag{5.5}$$

where R_s, $f(\boldsymbol{x})$, $C(\boldsymbol{x})$, and $\boldsymbol{x}$ are the system reliability, reliability function, cost function, and a set of design variables, respectively. Each design variable has a minimum and maximum redundancy value. C_0 is a given, fixed positive value for the cost constraint.

5.2.2 Multiobjective redundancy allocation optimization formulation

The multiobjective redundancy allocation optimization problem is expressed as a vector of functions:

$$\text{Minimize/maximize} \quad z = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_l(\boldsymbol{x})) \tag{5.6}$$

where z, l, $f_l(\boldsymbol{x})$, and $\boldsymbol{x}$ are the multiobjective vector function, the number of objective functions, the l-th objective function, and a set of design variables, respectively. In terms of minimization of all objectives, a feasible solution $\boldsymbol{x}_1$ is said to *dominate* another feasible solution $\boldsymbol{x}_2 (x_1 \succ x_2)$ if and only if $f_l(x_1) \le f_l(x_2)$ for $l = 1,...,L$ and $f_l(x_1) < f_l(x_2)$ for at least one objective function l [166]. A solution is said to be *Pareto optimal* if it is not dominated by any other solution in the solution space. The set of all such feasible non-dominated solutions in a solution space is termed the *Pareto optimal solution set*. For a given *Pareto optimal solution set*, the curve made in the objective space is called the *Pareto*

front. Since the number of Pareto optimal solutions is large and unknown for redundancy allocation problems, identifying the best known Pareto set by using a suitable MO optimization algorithm is a major challenge.

A two-objective redundancy allocation optimization problem to maximize reliability and minimize cost can be expressed mathematically as follows.

$$\text{Maximize} \quad f_1(\boldsymbol{x}) = f(\boldsymbol{x}) \tag{5.7}$$

$$\text{Minimize} \quad f_2(\boldsymbol{x}) = C(\boldsymbol{x}) \tag{5.8}$$

With such MO optimization problems, it is impossible to find a single optimum solution that optimizes both objective functions simultaneously in a MRAOP. Therefore, unlike the aim of finding the best solution to a SO optimization problem, the aim of optimizing two conflicting objectives simultaneously is to find a set of feasible solutions, each of which is not dominated by any other solutions.

5.3 Multiobjective hierarchical genetic algorithms

5.3.1 Hierarchical genetic algorithm

A Hierarchical Genetic Algorithm proposed by Yoshimura and Izui[158] is an advanced genetic algorithm that can represent hierarchical relationships among design variables using hierarchical genotypes, and can optimize hierarchical problems in a single optimization process. The term hierarchical genetic algorithm (HGA) comes from the use of a hierarchical approach when adapting a conventional GA[167]. While conventional genetic algorithms[122] use vector genotype structures, the HGA employs hierarchical genotype structures. The HGA is based on the fact that the conventional coding schemes for one dimensional arrays or even multi-dimensional arrays are not suitable for expressing design problems having hierarchical design structures. Moreover, vector coding scheme actually reduces the feasible design region as it uses artificial transformation of hierarchical design variables into vector form.

Note that, with hierarchical coding schemes, lower level genotype design variables depend upon upper level genotype design variables. When the value of an upper level genotype design variable changes, one or more lower level genotype design variables must also change, and the length of the genes may change. Since conventional genetic operators such as crossover and mutation operators cannot be applied to design variables expressed in hierarchical representations, new operators have been newly defined to handle hierarchical genotype encoding.

5.3.2 Multiobjective genetic algorithms

Single-objective genetic algorithms (GAs) that can be modified to solve MO optimization problems and find Pareto optimal sets in a single run are usually called multiobjective genetic algorithms (MOGAs)[168]. MOGAs are well suited to solving MO optimization problems because population-based approaches are applied and MOGAs can simultaneously search different parts of feasible design regions. Furthermore, difficult MO optimization problems that have discontinuous, non-convex, or multimodal solution spaces can be effectively solved by using customized MOGAs. Most of these MOGAs do not require artificial adjustments such as priority, scaling, or weighting coefficients for the objective functions[163]. An additional advantage is that the crossover and mutation operators may be modified to exploit the structural features of preferable solutions.

Over the years, a number of MOGAs have been developed and these can be broadly classified into two categories: elite MOGAs and non-elite MOGAs[163]. Non-elite MOGAs, as the name suggests, do not utilize elitism when selecting individuals for the next generation from the current population[169]. The first multi-objective GA, termed a vector evaluated GA (VEGA) proposed by Schaffer[170], was a non-elite MOGA. Other examples of important non-elite MOGAs are the Niched Pareto Genetic Algorithm (NPGA)[171], Weight-based Genetic Algorithms (WBGA)[172], Random Weighted Genetic Algorithms

(RWGA)[173], Non-dominated Sorting Genetic Algorithms (NSGA)[174], and Fonseca and Fleming's Multi-objective Genetic Algorithms (MOGA)[175].

In contrast, elite MOGAs employ an elite preservation operator in which an external set is created to store the best solutions of each generation. With this approach, the best individuals in each generation are preserved, and generated Pareto-fronts are close to the true Pareto front. Popular elite MOGAs include the Strength Pareto Evolutionary Algorithm (SPEA)[166], the improved SPEA (SPEA2)[165], the Pareto-Archived Evolution Strategy (PAES)[176], the Pareto Envelope-based Selection Algorithm (PESA)[177], Region-based Selection in Evolutionary Multiobjective Optimization (PESA-II)[178], the Fast Nondominated Sorting Genetic Algorithm (NSGA-II)[164], the Multi-objective Evolutionary Algorithm (MEA)[179], the Micro-GA[180], the Rank-Density Based Genetic Algorithm (RDGA)[181], and the Dynamic Multi-objective Evolutionary Algorithm (DMOEA)[182].

However, none of these MOGAs has been applied to the solving of MRAOPs that include hierarchical design variables. Though Yoshimura et al.[183] proposed a MO optimization method based on hierarchical arrangement of the design characteristics, its applicability is limited to machine product design. Additionally, the coding scheme proposed by Yoshimura and Izui[158] is not directly applicable to MRAOPs. The reason is that these problems contain hierarchical design variables with logical linkage, such as serial or parallel connections. Later, Kumar et al.[162] proposed a new coding scheme to encode the hierarchical design variables of MRAOPs and applied SO optimization using a HGA. Therefore, this paper formulates a new methodology for multiobjective optimization MRAOPs. In this methodology, a general framework for a multiobjective GA based on hierarchical genotype representation encoding for the design variables is proposed. As the proposed multiobjective genetic algorithm uses hierarchical approach in solution encoding, this is termed multiobjective hierarchical genetic algorithm (MOHGA). The following

section describes the common framework for MOHGAs.

5.3.3 Multiobjective hierarchical genetic algorithm

For handling hierarchical design variables and solving multiobjective MRAOPs, this paper proposes a general framework for multiobjective hierarchical genetic algorithm as follows:

Algorithm 1: Common framework for MOHGAs

Step 1: Initialize the population P in which each design variable is encoded by a hierarchical genotype

Step 2: Conduct a selection operation to select elite individuals from P and store their data in external set E (optional and not for non-elitist MOGAs)

Step 3: Create a mating pool using either P or E, or both

Step 5: Apply hierarchical crossover and hierarchical mutation operators

Step 6: Evaluate individuals

Step 7: Conduct reproduction based on the pool to create the next generation of P

Step 8: Combine P and E

Step 9: If termination criteria are not satisfied, return to Step 2.

In this general framework, selection of elite individuals can be conducted according to the user's designed algorithms while preserving the hierarchical genotype coding scheme. This paper implemented NSGA-II and SPEA2 for the selection operator in the above MOHGA. The rationale for implementing these two algorithms is that both of them include effective mechanisms for preserving diversity and can yield better Pareto optimal solution sets. Additionally, this paper applies SO using HGA and compares the optimal solutions with those obtained by MO using MOHGA with NSGA-II and SPEA2 at a certain fixed cost. This will aid in understanding the MRAOP search patterns to design more efficient algorithms. In the following subsections, solution encodings, selection algorithms, and

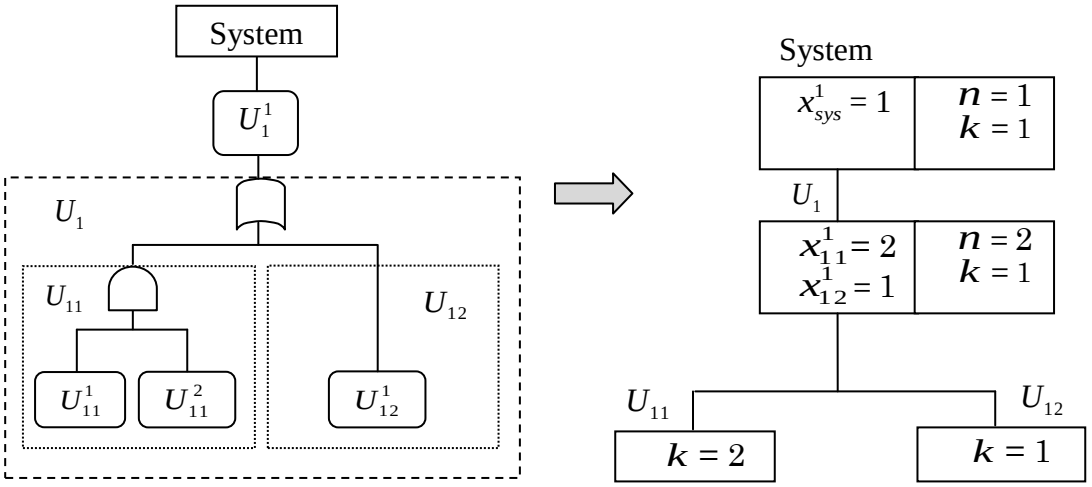
modified operators are described.

5.3.3.1 Solution encoding

The design variables in MRAOPs are encoded using hierarchical genotypes so that the hierarchical structure of these variables will be preserved intact. The hierarchical genotypes used here are represented using two types of nodes, termed ordinal and terminal nodes. Ordinal node N_i corresponds to redundancy unit U_i, and is characterized by several parameters and design variables. Parameters k_i and n_i stand for the redundancy of unit U_i, and the number of sub-units, respectively. Here, k_i is given by a design variable at an upper node, while the parameter n_i is a fixed value that depends on the optimization problems to be solved. $x_{i,m}^{j}$ is a design variable denoting the redundancy for the *m*-th sub-unit of the *j*-th redundancy unit, where j varies from 1 to k. Therefore, there are $n_i k_i$ design variables in unit U_i. A terminal node N_{t_i} corresponds to one of the lowest units, and incorporates design variable k_i, unit reliability r_i, and unit cost c_i. Since there are no sub-units, this terminal node does not contain parameter n_i or design variable $x_{i,m}^{j}$. Using these two genotype nodes, all possible optimal solutions for series reliability allocation problems can be represented.

Furthermore, the ordinal and terminal genotypes each have two evaluation functions, namely, reliability and cost. When the reliability function in the ordinal genotype is called, a calculation is conducted using Eq. (1). When calculating this equation, the reliability values of the lower units, $R_{i,m}^{j}$, are required, and these are obtained by calling the reliability function of the lower units. Finally, the reliability function of the terminal genotype returns its unit reliability r_i. Thus, the reliability functions are recursively called, and the total system reliability can be obtained. Similarly, the system cost can be obtained by calling the cost function embedded in each genotype.

The coding schemes used in the HGA can be understood more clearly by examining a redundancy allocation example for a two-level series unit U_1 having two sub-units, U_{11} and U_{12}, as shown in Fig.5.2.



(a) Redundancy allocation (b) Design variables at each node

Fig. 5.2. Hierarchical genotype representation in a bi-level series system U_1.

The redundancy values for sub-units U_{11} and U_{12} are 2 and 1, respectively. The HGA allows redundancy at two levels simultaneously. Fig.5.2(a) shows the redundancy allocation at both the system level and the level of the sub-units. Note that genotypes which use fixed arrays, frequently used in various optimization problems, are not applicable to MRAOPs since the number of design variables varies according to the number of redundant units. Here, the number of genes varies dynamically based on the proposed solution configuration, and the applied coding scheme is capable of handling dynamic variations in the values of design variables during the optimization process.

5.3.3.2 Hierarchical crossover

Crossover operations between individuals are conducted among each corresponding set of genes, using a two-step procedure. For the initial step, any other individual is first selected as the crossover partner and crossover operators then exchange the corresponding genes of the two individuals. Here, when a gene of an alternative for a substructure is

exchanged with the corresponding gene of another alternative, all corresponding lower substructures are also exchanged, to preserve consistency in the selection of alternatives. If this operation were not conducted in this way, meaningless lower structures might be generated in the lower positions of the exchanged substructures. The algorithmic procedures are follows:

Algorithm 4: Crossover for handling hierarchical genotype

Step 1: Select two individuals for crossover operations, then find the set of genes at the highest level of the multilevel structural system for each of the two individuals, and start the crossover operation with probability p_{c_1}.

Step 2.1: If the gene $x_{i,m}^{j}$ of individual 1 and that of individual 2 are different; conduct a crossover operation for $x_{i,m}^{j}$ with probability p_{c_2}. This operation is the same as a uniform crossover of simple genetic algorithms with p_{c_2} set to 0.5. Then, proceed to Step 2.3. If crossover operations are not conducted, proceed to Step 2.4. If the genes of both individuals are the same, proceed to Step 2.2.

Step 2.2: If $x_{i,m}^{j}$ contains a subordinate set of genes; it will be examined for possible crossover operations in Step 2.1. Otherwise, proceed to Step 2.4.

Step 2.3: When $x_{i,m}^{j}$ genes are exchanged between individuals 1 and 2, the lower substructures of each individual are also exchanged.

Step 2.4: Increment m by 1. When $m > n$, m =1, increment j by 1. When $j > k$, end the crossover operations since the set of genes has been exhausted, and return to the crossover operations for the parent set of genes.

Fig.5.3 shows an example of the crossover operation between unit U_{12} of the first individual and U_{12} of the second individual. With exchange of parent unit $U_{12,}$ the subunits

of U_{12} are also exchanged. Note that the design variables of U_{12} of both the parents are unequal in the given example. However, if the design variables of U_{12} in both the parents are equal, the subunits of U_{12} are subject to crossover operation.

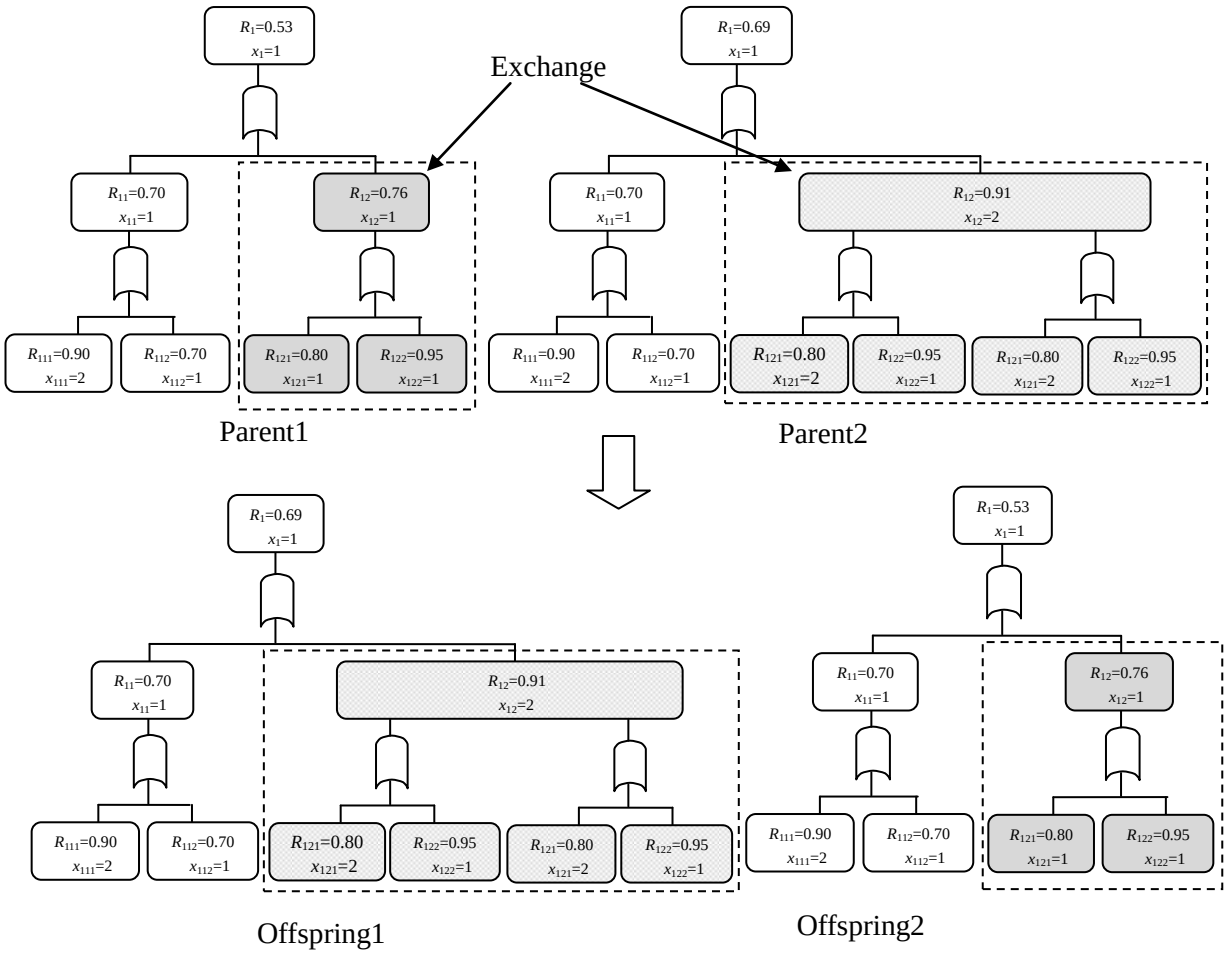


Fig.5.3. An example of hierarchical crossover operation in a 3-level series system U_1

5.3.3.3 Hierarchical mutation

In mutation operations, mutation operators are first applied to the set of genes at the highest level of the multilevel structural system, and mutation operators are recursively applied to the sets at sub-unit levels in the same way as for crossover operators. The algorithmic procedures are as follows:

Algorithm 5: Mutation for handling hierarchical genotype

Step 1: Examine the substructure at the highest level.

Step 2.1: Determine whether or not a mutation operation should be conducted, with mutation probability p_m for the gene $x_{i,m}^{j}$. If the mutation is conducted, proceed to Step 2.3 otherwise proceed to Step 2.2.

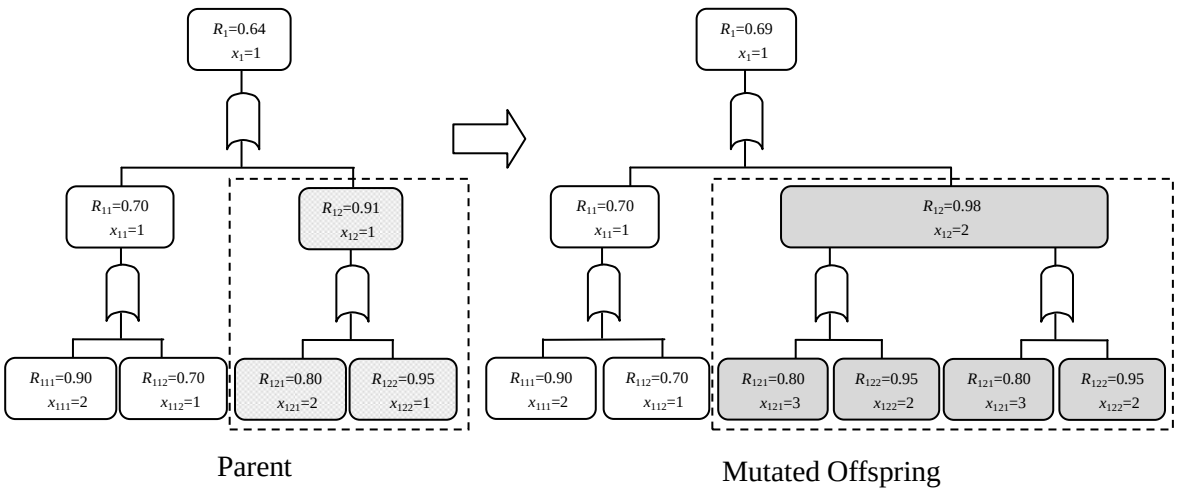
Step 2.2: If $x_{i,m}^{j}$ contains set of genes at sub-unit levels, proceed to Step 2.1 and examine sub-unit set of genes. If not, proceed to Step 2.5.

Step 2.3: Randomly generate $x_{i,m}^{j}$.

Step 2.4: Randomly reconstruct the genes of all sub-unit's node for the selected alternative.

Step 2.5: Increment m by 1. When $m > n$, m =1, increment j by 1. When $j > k$, end the crossover operations since the set of genes has been exhausted, and return to the crossover operations for the parent set of genes.

Fig.5.4 shows an example of the mutation operation in a unit U_{12} of an individual U_1. A new structure of unit U_{12} along with its subunits is randomly generated and replaces the older configuration of U_{12}. The design variables of lower units of U_{12} are also randomly generated.



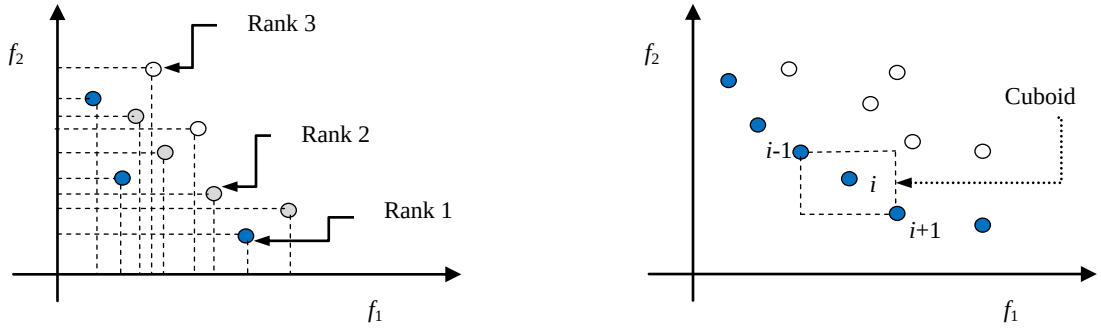
Parent Mutated Offspring

Fig. 5.4. An example of hierarchical mutation operation in a 3-level series system U_1

5.3.3.4 Selection operator

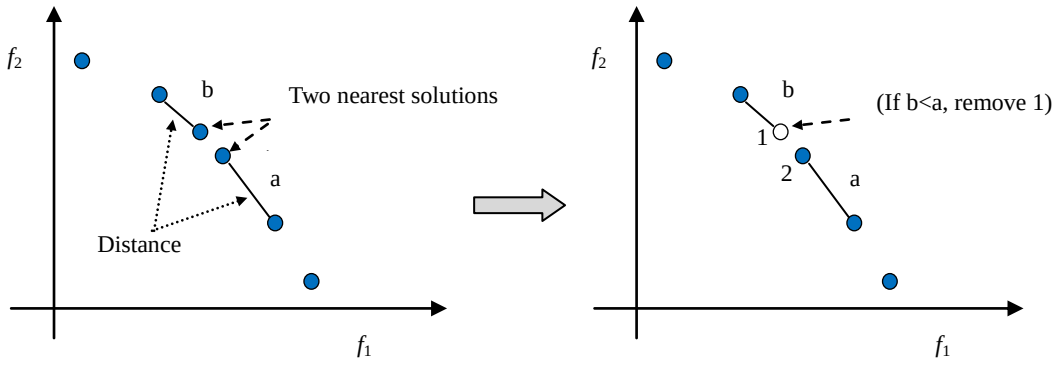
The selection method used in this paper implements NSGA-II and SPEA2 algorithms.

Though based on different principles, they both have excellent mechanism for the preservation of diversity.



(a) Non-dominated ranking (b) The crowding distance calculation

Fig. 5.5. Ranking and crowding distance concepts used in NSGA-II



(a) Two nearest solutions on Pareto front and distance ‘a’ and ‘b’ are compared

(b) Since ‘b’ is less than ‘a’, solution ‘1’ is removed

Fig. 5.6. Truncation Operator used in SPEA2

Fig.5.5 and Fig.5.6 illustrate the different concepts, the ranking and crowding distance used in NSGA-II, and the truncation operator used in SPEA2. These algorithms can be integrated within the general framework of hierarchical genetic algorithms, for which details are given in the following subsections.

5.3.3.4.1 NSGA-II

NSGA-II is a fast, elite MOGA proposed by Deb et al.[164]. The complete procedure that NSGA-II uses within the proposed hierarchical framework is given below.

Algorithm 2: NSGA-II

Step 1: Create a random population P_0 of size N_P in which each design variable is encoded by a hierarchical genotype. Set $t = 0$

Step 2: Apply hierarchical crossover and mutation operators to P_0 to create an offspring population Q_0 of size $N_Q = N_P$

Step 3: If the stop criterion is satisfied, stop and return P_t as output.

Step 4: Set $S_t = P_t \cup Q_t$, apply a non-dominated sorting algorithm and identify different fronts F_1, $F_2, \ldots, F_k$.

Step 6: Set new population P_{t+1} =0. Set counter i = 1. Until $|P_{t+1}| + |F_i| \leq N$ set $P_{t+1} = P_{t+1} \cup F_i$ and $i = i + 1$.

Step 7: Perform the crowding-sort procedure and include the most widely spread $(N - |P_{t+1}|)$ solutions found using the crowding distance values in sorted F_i in P_{t+1}.

Step 8: Apply the crowded tournament selection, hierarchical crossover and mutation operators to P_{t+1} to create offspring population Q_{t+1}.

Step 8: Set $t = t + 1$, then return to Step 3.

Note that when the combined parent and offspring population includes more than N_P non-dominated solutions, NSGA-II acts as a pure elitist GA where only nondominated solutions participate in crossover and selection.

5.3.3.4.2 SPEA2

SPEA[166] and SPEA2[167] are both very effective algorithms that use an external list to store non-dominated solutions discovered in the course of searching. They are also excellent examples for the use of external populations. The procedure for using SPEA2 within the proposed hierarchical framework is as follows:

Algorithm 3: SPEA2

Step 1: Create a random population P_0 of size N_P in which each design variable is encoded by a hierarchical genotype. Set $t = 0$ and an empty external archive E_0 of size N_E

Step 2: Calculate the fitness of each solution ***x*** in $P_t \cup E_t$ as follows:

Step 2.1: Calculate the raw fitness as $R(x,t) = \sum_{y \in P_t UE_t, y \succ x} S(y,t)$ where $S(y,t)$ is the number of solutions in $P_t \cup E_t$ dominated by solution y.

Step 2.2: Calculate the density as $D(x,t) = (\sigma_x^k + 2)^{-1}$ where σ_x^k is the distance between solution *x* and its *k*-th nearest neighbor, where $k = \sqrt{N_P + N_E}$

Step 2.3: Assign a fitness value as $F(x,t) = R(x,t) + D(x,t)$.

Step 3: Copy all non-dominated solutions in $P_t \cup E_t$ to E_{t+1}. Now, two cases may arise. Case 1: If $|E_{t+1}| > N_E$, then truncate $|E_{t+1}| - N_E$ solutions by iteratively removing solutions that have maximum σ^k distances. Break any tie by examining σ^l for l =k-1,…, 1 sequentially. Case 2: If $|E_{t+1}| \le N_E$, copy the best $N_E - |E_{t+1}|$ dominated solutions according to their fitness values from $P_t \cup E_t$ to E_{t+1}.

Step 4: If the stopping criterion is satisfied, stop and copy non-dominated solutions in E_{t+1}.

Step 5: Select parent from E_{t+1} using binary tournament selection with replacement.

Step 6: Apply hierarchical crossover and mutation operator to the parents to create N offspring solutions. Copy offspring to $P_{t+1}, t = t+1$, then return to Step 2.

5.4 Numerical examples

In this section, two MRAOPs with different multilevel structures are solved by applying MO optimization using proposed MOHGA and SO optimization using a HGA, respectively. In the latter problem, the objective function is to maximize the system

reliability subject to a cost constraint. On the other hand, in the MO optimization, the objective functions are the maximization of system reliability and minimization of system cost. Further, since the feasible design space is very large in MRAOPs, the efficiency of both genotype selection and the search for optimal solutions is analyzed for both problems, to pinpoint the most viable approach for multilevel redundancy allocation optimization problems.

5.4.1 Problems

Fig.5.7 and Fig.5.8 show the two multilevel structures. The reliability of these systems can be maximized by allocating appropriate redundancy to their units located at different levels. In the first problem, *Problem-A*, there are three hierarchical levels, whereas in the second problem, *Problem-B*, has four levels. All units at all levels of both the problems are serially connected. It can be seen that *Problem-A* structures are not symmetrical.

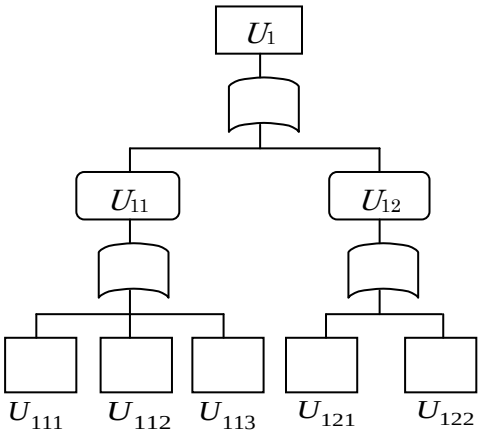


Fig. 5.7. *Problem-A* (a three level multilevel series system).

The reliability of both problems is calculated using (5.1) and (5.2). The reliability for *Problem-A* is expressed as:

$$R_s = 1-(1-R_1)^{x_1} = 1-(1-\prod_{i=1}^{2}[1-(1-R_{1i})^{x_{1i}}])^{x_1} = 1-(1-\prod_{i=1}^{2}[1-(1-\prod_{j=1}^{3}[1-(1-R_{1ij})^{x_{1ij}}])^{x_{1i}}])^{x_1} \quad (5.9)$$

The design variables x_i also have given maximum and minimum values. A similar

reliability expression can be obtained for *Problem-B.*

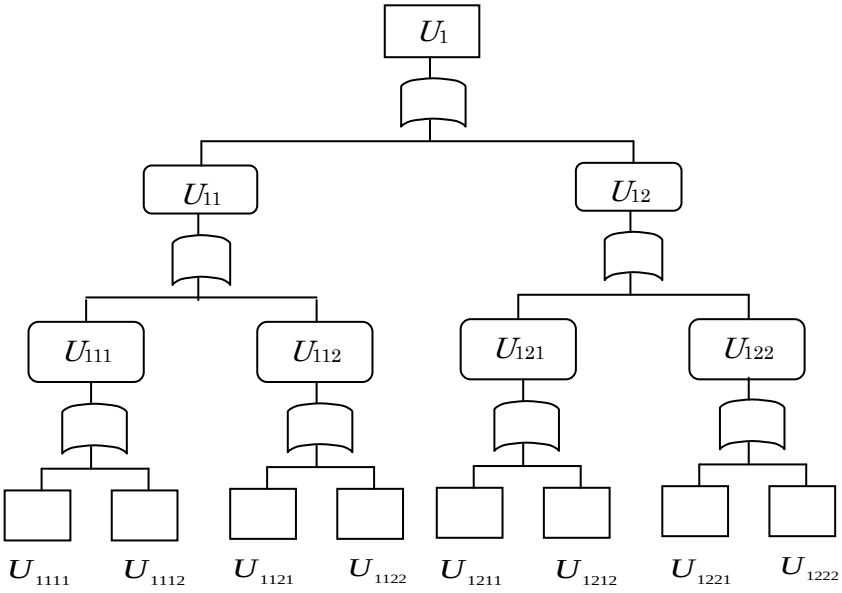


Fig. 5.8. *Problem-B* (a four level multilevel series system).

5.4.2 Input data

The HGA parameters are based on several experimental runs conducted for both problems. Table 5.1 and Table 5.2 summarize the best parameters and corresponding results obtained in 10 trials of 1000 generations using 100 individuals. The best values for crossover and mutation probabilities are 0.8 and 0.05, respectively. Similarly, to find the best genetic parameter values for MOHGA with NSGA-II and SPEA2, several Pareto fronts were obtained in 10 trials of 1000 generations using 100 individuals. The results indicate that the best crossover and mutation probabilities are 0.9 and 0.01, respectively.

NSGA-II and SPEA2 were both assigned a population size of 100 individuals based on several trial and error evaluations. The HGA for the SO problem were also assigned the same population size. Note that the archive size in MOHGA with SPEA2 was set to 100 after evaluating the performance with different archive sizes in experimental runs. Both the archive in MOHGA with SPEA2 and the offspring size used in MOHGA with NSGA-II have a value of 100, increasing the utility of comparisons between these two algorithms.

The number of generations for HGA, MOHGA with SPEA2, and MOHGA with NSGA-II were set to 1000, a figure obtained through trial-and-error analysis.

TABLE 5.1

HGA PARAMETERS

Cases	Parameters		Average Fitness	Best Fitness
	Crossover	Mutation	(20-runs)	(20-runs)
1	0.7	0.05	0.96047	0.97628
2	0.9	0.05	0.96155	0.97628
3	0.8	0.05	0.9621	0.97639
4	1.0	0.05	0.96117	0.97628
5	0.8	0.01	0.92826	0.97254
6	0.8	0.1	0.94484	0.96422
7	0.8	0.2	0.931904	0.950827

TABLE 5.2

MOGA PARAMETERS

MOGA	Crossover	Mutation	Population
NSGA-II	0.9	0.01	100
SPEA2	0.9	0.01	100

The number of units and the corresponding fixed reliability values for *Problem-A* and *Problem-B* are 8 and 15, respectively. A maximum redundancy number of five was imposed for both the problems. Table 5.3 summarizes the unit reliability and unit cost of the components at the very lowest level in both problems. The unit reliability and the unit cost at the very lowest level in the multilevel redundancy allocation problems were used when calculating the unit reliability and the unit cost of upper level units, up to the system level.

TABLE 5.3
INPUT DATA

Problem-A				*Problem-B*			
Unit	Reliability	Cost	λ	Unit	Reliability	Cost	λ
U_1	0.3591	51	2	U_1	0.2085	102	2
U_{11}	0.5700	29	2	U_{11}	0.4275	48	2
U_{12}	0.6300	18	2	U_{12}	0.4877	50	2
U_{111}	0.8000	5	3	U_{111}	0.6000	21	3
U_{112}	0.7500	9	3	U_{112}	0.7125	21	3
U_{113}	0.9500	6	3	U_{121}	0.7650	23	3
U_{121}	0.7000	5	3	U_{122}	0.6375	21	3
U_{122}	0.9000	7	3	U_{1111}	0.7500	7	4
				U_{1112}	0.8000	6	4
				U_{1121}	0.7500	8	4
				U_{1122}	0.9500	5	4
				U_{1211}	0.9000	9	4
				U_{1212}	0.8500	6	4
				U_{1221}	0.7500	5	4
				U_{1222}	0.8500	8	4

5.4.3 Computational results

In solving *Problem-A* and *Problem-B*, 10 trials of 1000 generations were conducted for all the cases. For the SO problem, the best solution among the 10 trials was chosen as the final optimum solution. Similarly, the best Pareto front among the 10 trials of 1000 generations was selected as the final solution for the two-objective problem. Fig. 5.9, Fig. 5.10, Fig. 5.11, and Fig. 5.12 show plots of Pareto optimal solution sets obtained by the MO optimization using MOHGA with NSGA-II and MOHGA with SPEA2 when solving *Problem-A* and *Problem-B*. The optimal solutions by for the SO optimization using the HGA subject to a cost constraint of 500 are also obtained for the two problems.

In Fig.5.9, the Pareto front obtained using MOHGA with SPEA2 is dominated by

the MOHGA with NSGA-II solutions. The best solution of the SO optimization using the HGA is also plotted in Fig.5.9 and this solution to a 3-level series problem dominates all Pareto optimal solutions obtained by the MO optimization using MOHGA with NSGA-II and MOHGA with SPEA2. For the 4-level problem shown in Fig.5.11, the optimal Pareto front obtained using MOHGA with NSGA-II is superior to that using MOHGA with SPEA2 and the optimal solutions obtained by the SO optimization using the HGA dominate all the optimal solutions of the Pareto front obtained using MOHGA with SPEA2, but are inferior to those using NSGA-II.

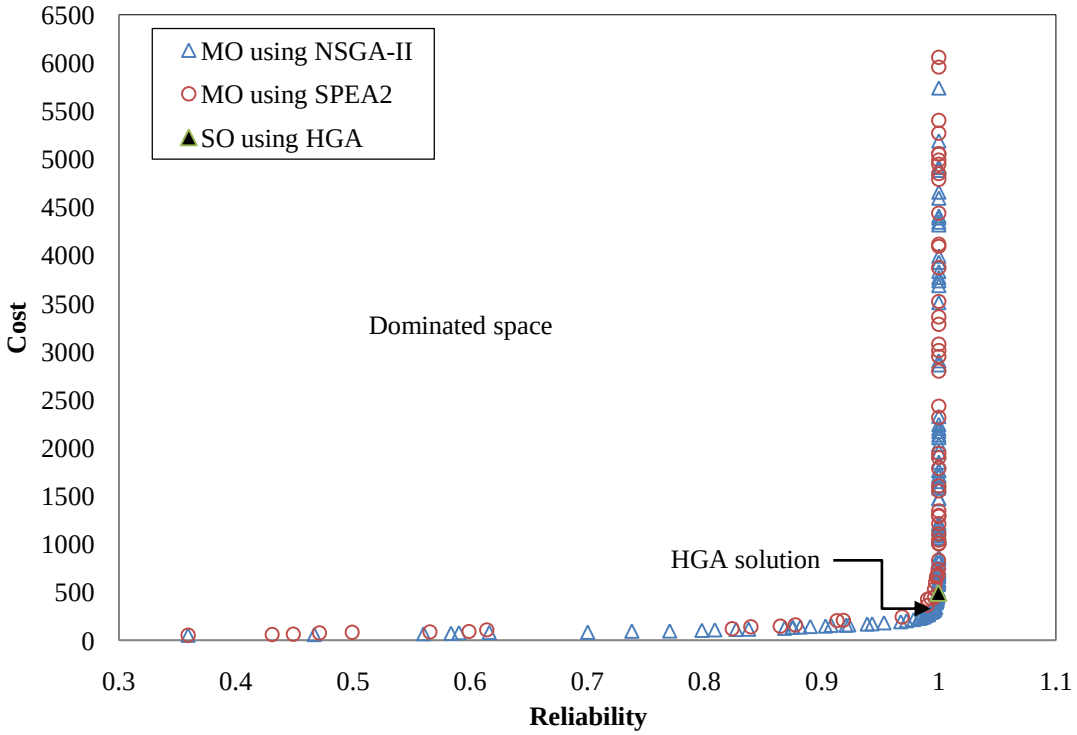


Fig. 5.9. Redundancy allocation in *Problem-A* using NSGA-II, SPEA2, and HGA

Fig. 5.10 and Fig.5.12 show enlarged plots of the optimal solutions obtained in the MO optimizations using MOHGA with NSGA-II and MOHGA with SPEA2 and the SO optimization using the HGA for both the problems. As shown in these figures, the performance of MOHGA with SPEA2 in 4-level problems deteriorated compared to the results for the 3-level problem. In other words, the size of the search space affected the convergence of optimal solutions when using MOHGA with SPEA2.

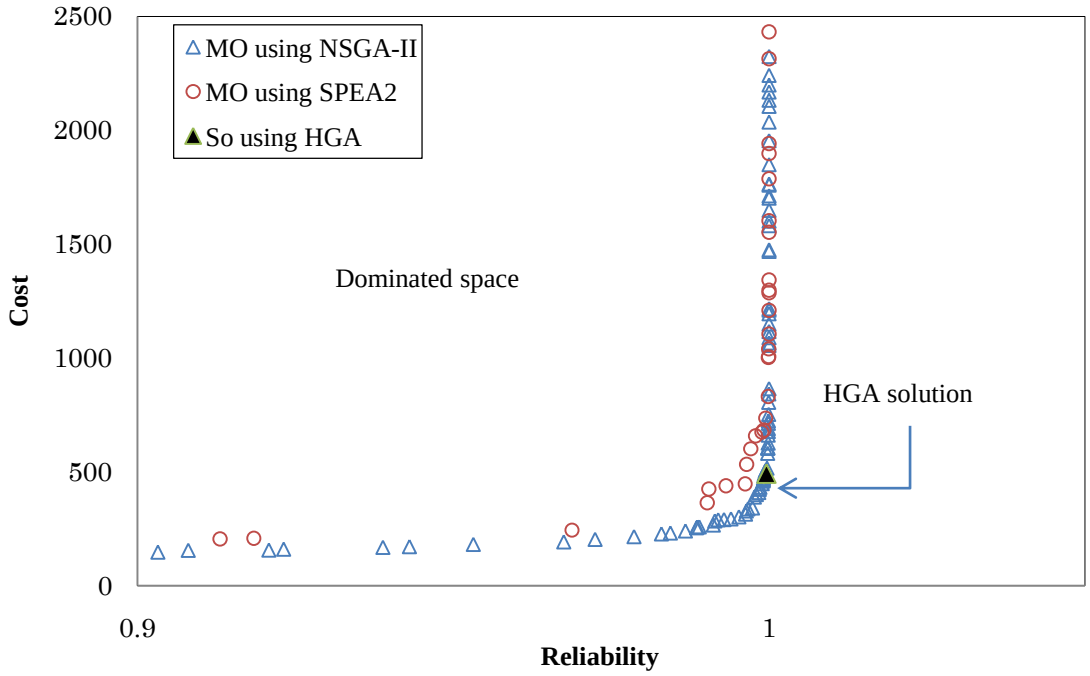


Fig. 5.10. Enlarged view of the Pareto front in *Problem-A*

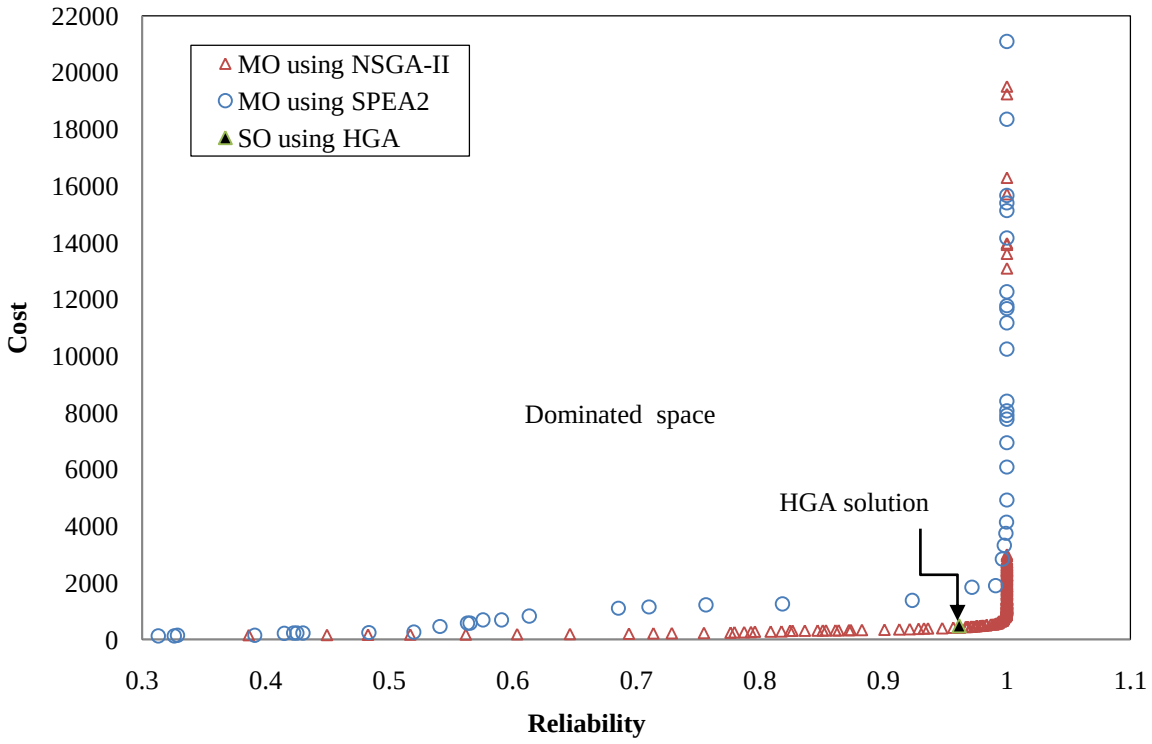


Fig. 5.11. Redundancy allocation in *Problem-B* using NSGA-II, SPEA2, and HGA

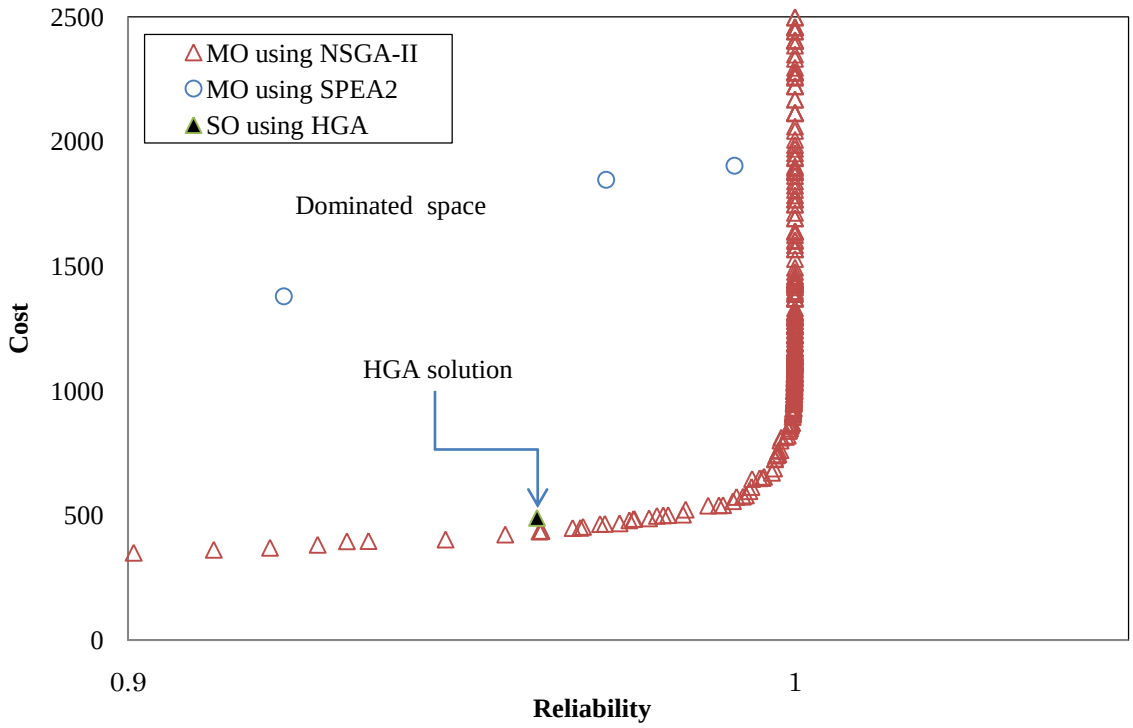


Fig. 5.12. Enlarged view of the Pareto front in *Problem-B*

The inferior performance of MOHGA with SPEA2 in both problems is analyzed further in terms of Pareto front movement as a function of the number of generations during optimization. Fig. 5.13 and Fig. 5.14 show the movements of the Pareto fronts in the 4-level problem. The figures show three Pareto fronts after 50 generations of the first 1000-generation trial, after 1000 generations of the first 1000-generation trial, and after 10000 generations, at the end of the tenth 1000-generation trial. The search direction in both the algorithms is clearly visible. With SPEA2, the search direction is from high cost to low cost regions, while maintaining several extreme solutions on each generation's Pareto front. In contrast, the NSGA-II Pareto front moves towards the low cost region without preserving each generation's extreme solutions. Instead, the entire Pareto front shifts as new solution sets are obtained. In other words, MOHGA with SPEA2 yields Pareto fronts with wider spans or diversity, while MOHGA with NSGA-II distributes solutions on Pareto fronts in a more focused manner.

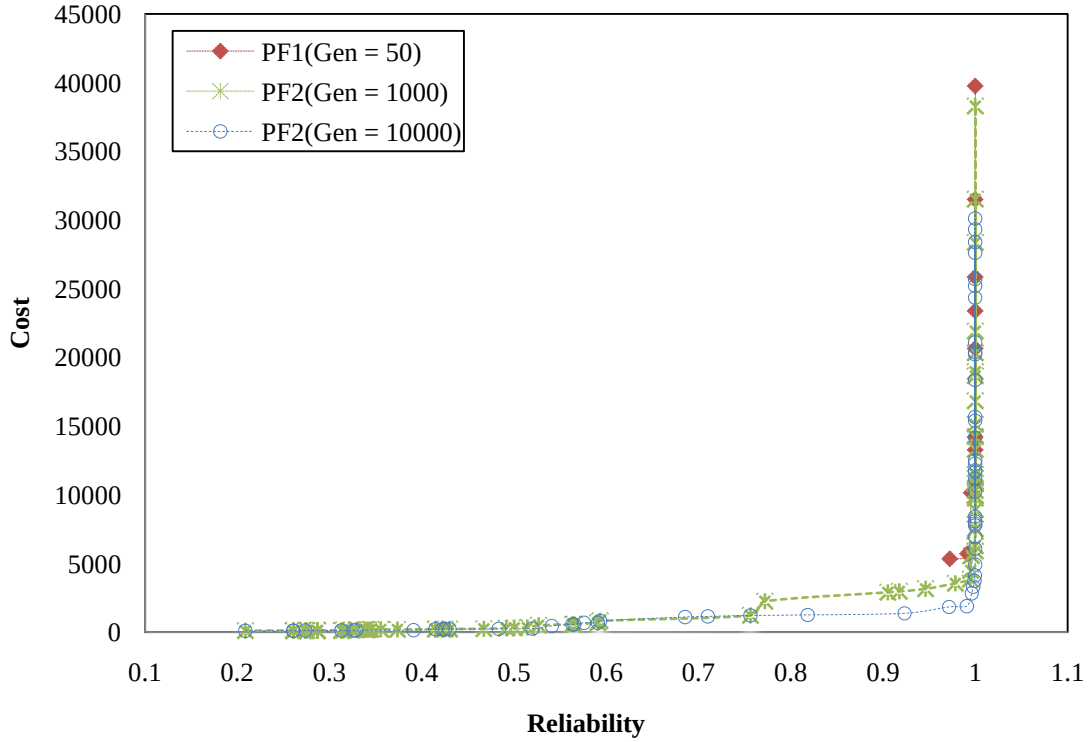


Fig. 5.13. Pareto front movement in MO of *Problem-B* using SPEA2

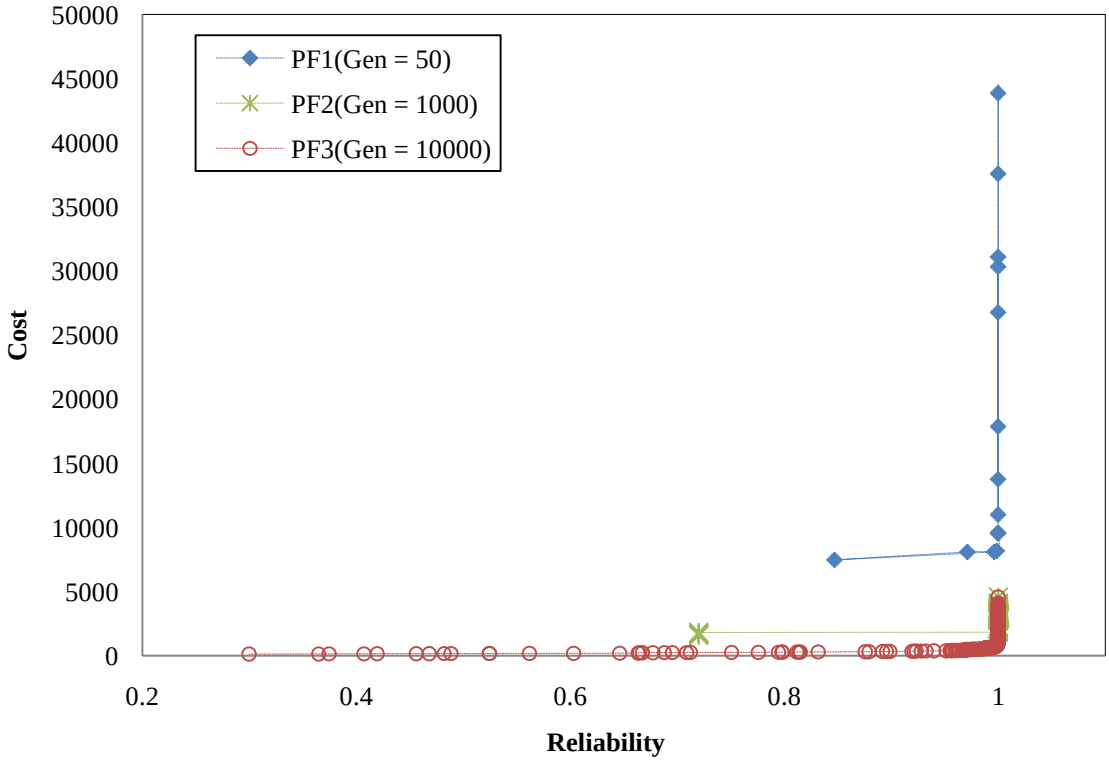


Fig. 5.14. Pareto front movement in MO of *Problem-B* using NSGA-II

Similarly, Fig. 5.15 and Fig. 5.16 show population distributions when using MOHGA

with SPEA2 and MOHGA with NSGA-II during optimization after 50 generations of the first 1000-generation trial, after 1000 generations of the first 1000-generation trial, and after 10000 generations, at the end of the tenth 1000-generation trial

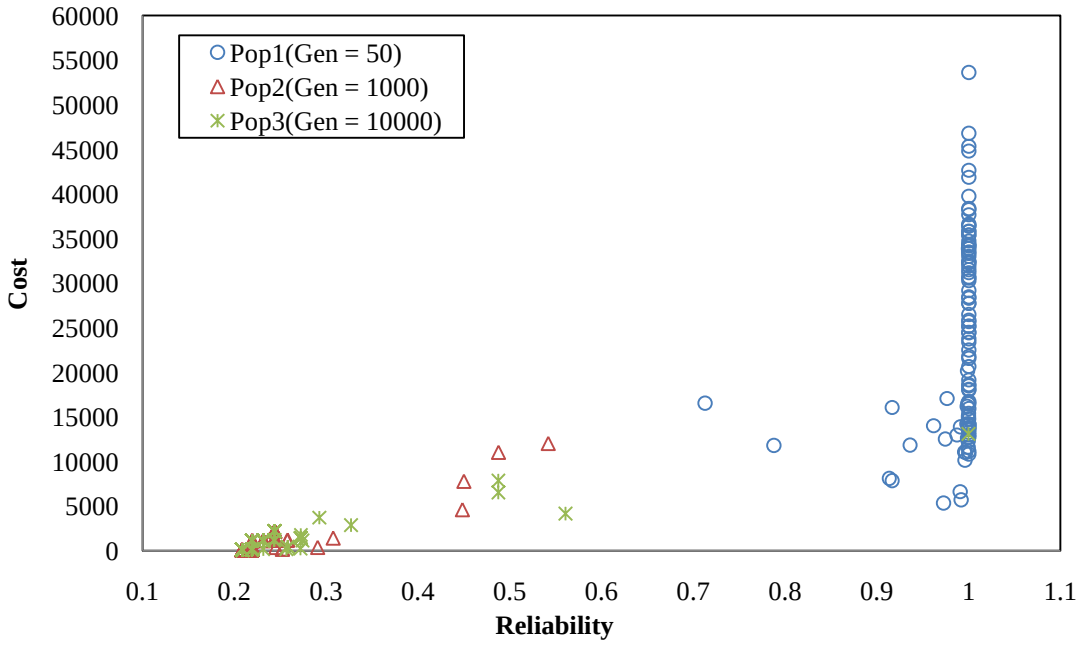


Fig. 5.15. Population distribution in MO of *Problem-B* using SPEA2

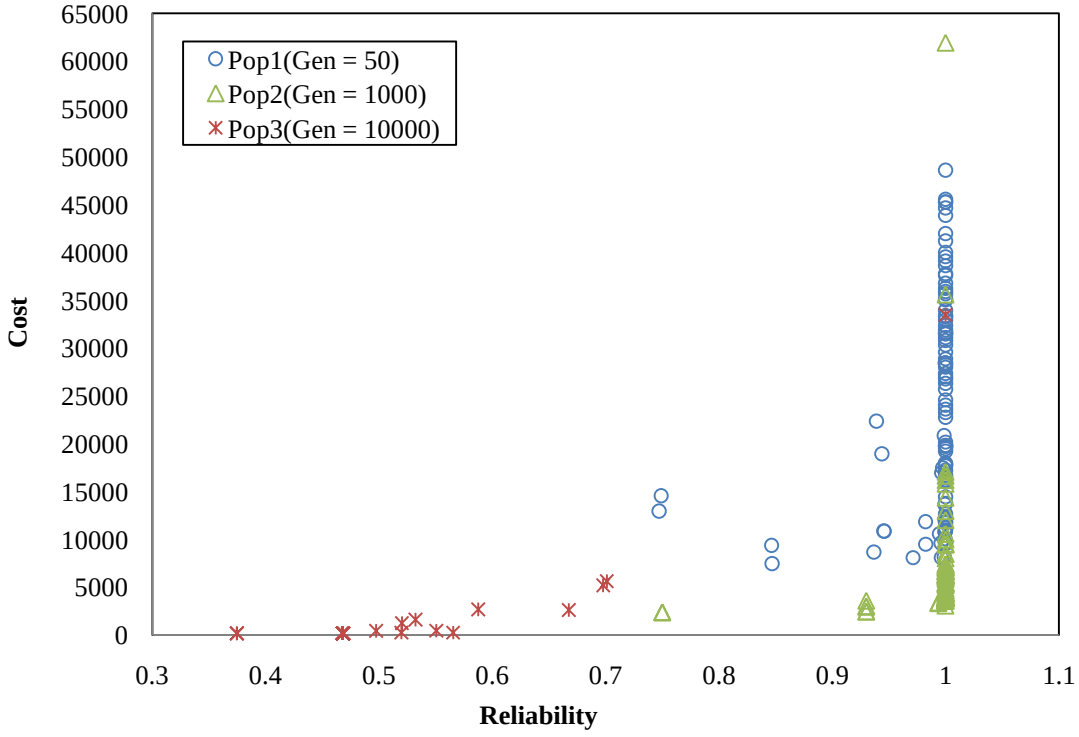
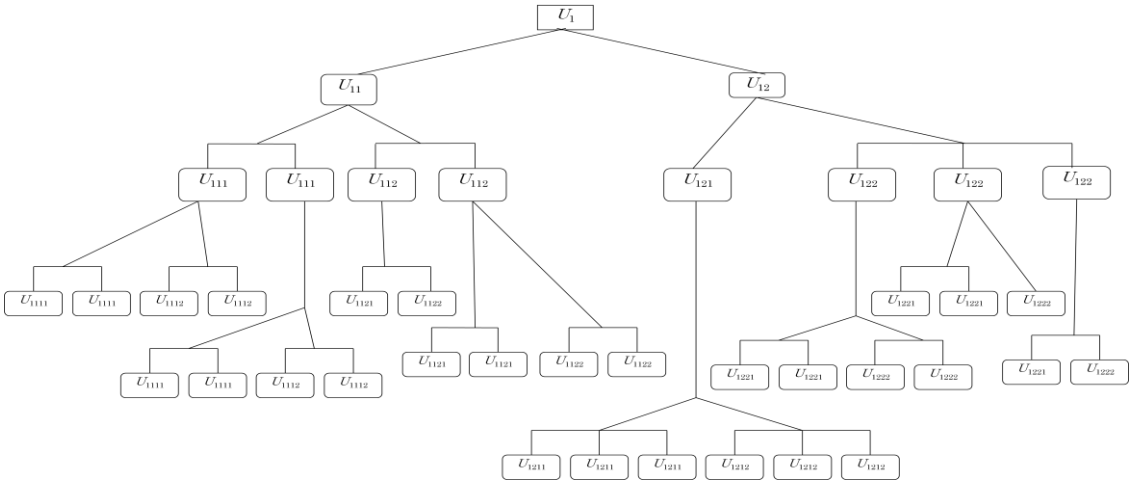
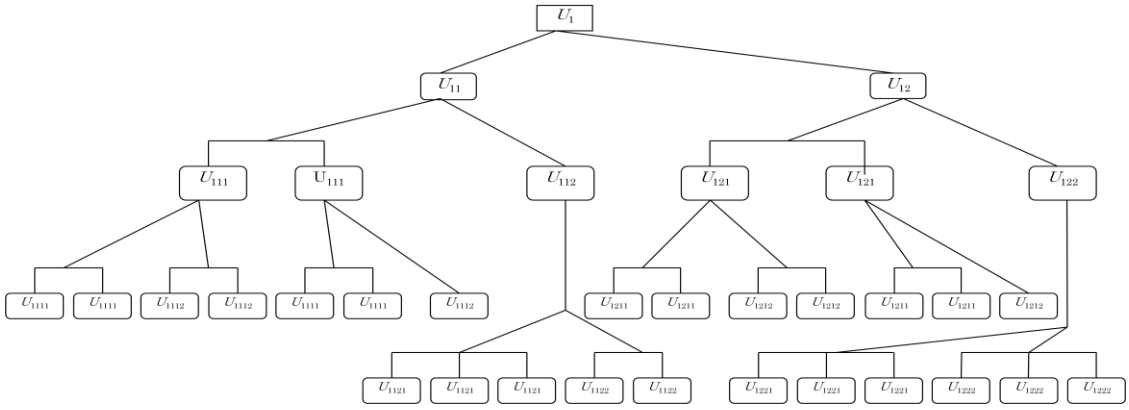


Fig. 5.16. Population distribution in MO of *Problem-B* using NSGA-II



(a) The best solution is R_s = 0.96134

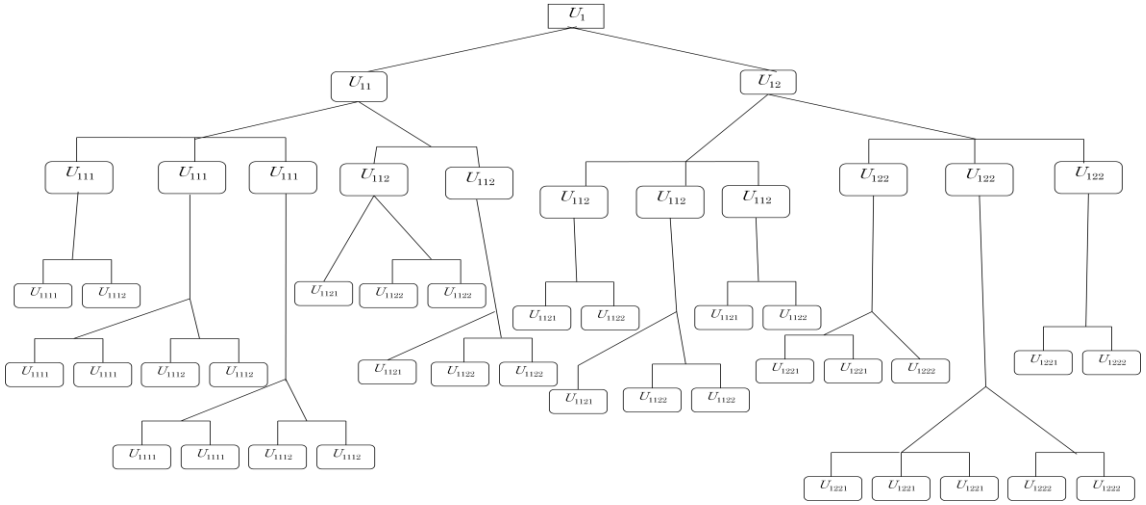


(b) The second best solution is R_s = 0.93355

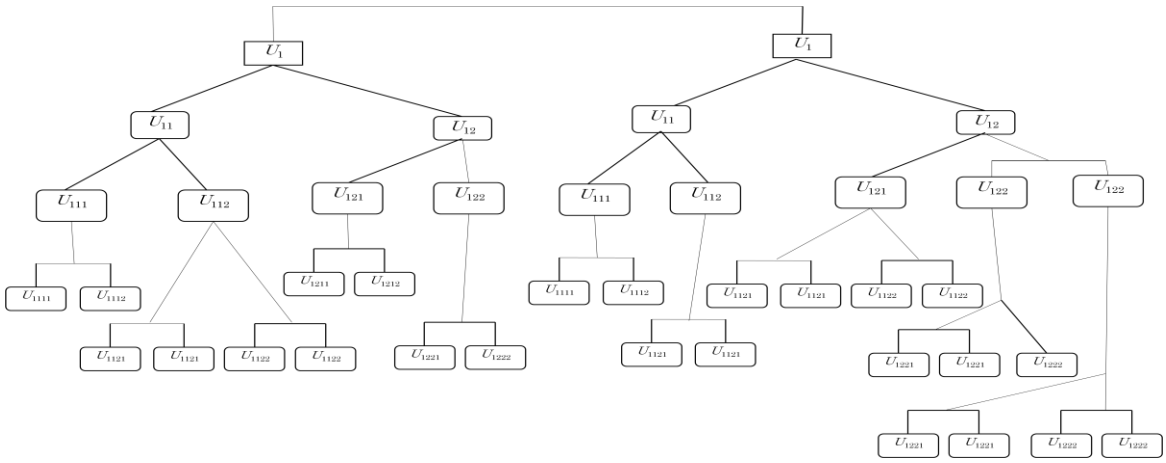
Fig. 5.17. Optimal structures obtained in SO using HGA

To compare the three optimization strategies, a single solution is chosen from solution sets where the cost constraint is 500. Fig. 5.17 shows the optimal structures obtained using single objective optimization using HGA and Fig.5.18 shows those obtained using MOHGA with NSGA-II, and MOHGA with SPEA2 when solving the 4-level problem. Fig. 5.17 (a) shows the best optimal structure with optimal reliability of 0.9613 and Fig.5.17(b) the second best solution with reliability 0.93335 obtained using the HGA. Similarly, the optimal structures obtained using MOHGA with NSGA-II and MOHGA with SPEA2 are shown in Fig. 5.18 (a) and (b). The optimal reliabilities obtained using MOHGA with NSGA-II and MOHGA with SPEA2 are 0.9748 and 0.5690, respectively. If

we look at these figures we find that the MOHGA with NSGA-II algorithm yielded superior Pareto solution sets compared to MOHGA with SPEA2.



(a) R_s = 0.97483 with NSGA-II



(b) R_s = 0.56903 with SPEA2

Fig. 5.18. Optimal structures obtained in MO using MOHGA

5.5 Discussion

In multilevel redundancy allocation optimization, the basic structure of the problems is kept intact when encoding the design variables. During the optimization process, the values of design variables expressing redundancy are interdependent, since they are hierarchical, i.e., the reliability of an upper level unit depends on the reliability of its lower level units. The results obtained here show that optimizations using hierarchical genotype encoding

have advantages in finding optimal structures when compared with conventional vector-type encoding schemes. The performance of a given optimization method may be greatly affected by the number of levels and the distribution of units in the basic structure of the MRAOP. This paper examined the performances of a SO method using a HGA and a MO method using MOHGA with NSGA-II and SPEA2.

The results of the numerical examples demonstrate that MO optimizations are better at solving multilevel redundancy allocation problems than SO techniques. MO optimizations provide Pareto optimal solution sets in a single run, and offer decision-makers a wider range of choices from which to choose the best solution for the specific problem at hand. Moreover, MO methods using MOHGA with NSGA-II and MOHGA with SPEA2 include schemes for preserving good population diversity when solving MRAOPs and, for large-scale problems, can provide better solutions than SO optimizations, as the results of the provided numerical examples also demonstrate.

We observe that in MRAOPs, the performance of the MOHGA with NSGA-II in obtaining a set of Pareto front is superior to that of MOHGA with SPEA2, however the SO optimization using a HGA for the 3-level problem yielded a better solution than either MOHGA with NSGA-II or MOHGA with SPEA2 applied in a MO optimization with a system cost constraint of 500. For the 4-level problem, MOHGA with NSGA-II provided a better solution than MOHGA with SPEA2 or the HGA under similar settings. These results show that the size of the search space can drastically affect the performance of different algorithms, hence choosing the most suitable algorithm when attempting to solve a particular MRAOP is extremely important.

As shown in Fig.5.17 and Fig.5.18, the optimal structures for the *Problem-B*, at the cost of 500, obtained in SO using the HGA and in MO using the MOHGA with NSGA-II and SPEA2 all have different arrangements of units. The best among all optimal structures

is obtained by MOHGA with NSGA-II. Examining the structures, we find that the number of redundant units in the second and the third levels most significantly affects the reliability of this solution. In other words, the solution obtained with MOHGA with NSGA-II has the highest ratio of the third level's units to the second level's units among all solutions provided by either MO using MOHGA with SPEA2 or SO using HGA. Additionally, the ratio of third level's units to second level's units provided by the MOHGA with SPEA2 is the smallest among all methods used here.

The observed difference in the performance of the SO optimization using a HGA and a MO optimization using MOHGA with NSGA-II or SPEA2 can be attributed to the diversity preservation mechanism that is not available in the SO using HGA. In SO using HGA, redundancy allocation at module levels often violates the cost constraint and thus the HGA fails to preserve the solutions with more number of redundancies at middle levels. Concerning the observed poor performance of MOHGA with SPEA2 in comparison to MOHGA with NSGA-II when solving the MRAOPs here, this is due to the difference selection strategies used. In MOHGA with NSGA-II, dominance ranking is used when forming the fronts of individuals and these fronts are first used to populate the external set, based on ranking, a strategy that allows a set of close-neighbor individuals in the same front to be included in the next generation. In contrast, the MOHGA with SPEA2 selects individuals according to assigned fitness values based on Euclidean density information, so close-neighbor individuals are likely to be excluded in the next generation. The MOHGA with SPEA2 therefore yields Pareto fronts showing wider distributions of non-dominated solutions, whereas the MOHGA with NSGA-II is more focused when exploring the search space and generating Pareto solution sets. The reason for this behavior is that MOHGA with SPEA2 uses a truncation operator based on a nearest-neighbor strategy, but MOHGA with NSGA-II uses crowding distance when the size of non-dominated solutions exceed the

archive size. Fig.5.13 and Fig.5.14 show that MOHGA with NSGA-II discards non-dominated solutions in the high cost region since the crowding distance there tends to have lower values. On the other hand, low cost non-dominated solutions easily survive to the next generation, because the crowding distance of such solutions is high.

5.6 Summary

This chapter proposed the multiobjective formulation of multilevel redundancy allocation problems and multiobjective hierarchical genetic algorithms (MOHGAs) to solve them. For the numerical examples, two multilevel redundancy allocation problems having 3 and 4 hierarchical levels were solved to maximize reliability while minimizing cost. The proposed MOHGAs were applied to solve the MRAOPs. The selection operators of MOHGAs used NSGA-II and SPEA2. Also, for comparison with MO using MOHGA with NSGA-II and SPEA2 results, a SO optimization using a HGA was also applied to solve the MRAOPs.

The results show that, for large multilevel problems, MO optimizations using MOHGA with NSGA-II are preferable to a SO optimization using a HGA because they provide superior solutions, as well as a Pareto set of optimal solutions that can be used during subsequent decision making. Additionally, the results show that the MOHGA with NSGA-II is superior to the MOHGA with SPEA2 for solving MRAOPs. Building on the research carried out for this work, the authors hope to design improved selection strategies for multiobjective HGAs which can effectively preserve the structural diversity of individuals when searching for optimal solutions to large-scale MRAOPs.

Chapter 6

Conclusions and future works

This chapter presents an overview and general conclusions related to the work developed in this dissertation. The general topic of research is developing novel methodologies for optimal reliability design of hierarchical systems. In chapters 2, hierarchical and modular concepts in redundancy allocation is presented and a general formulation of multilevel redundancy allocation problems is proposed. Chapter 3 developed an innovative hierarchical genetic algorithm for solving reliability optimization problems for hierarchical systems. Multilevel redundancy allocation optimization problems involve hierarchical design variables. Conventional genetic algorithm uses vector representation for encoding design variables and yield suboptimal solutions. Therefore, the main objectives of chapter 3 were to represent the exact structure of hierarchical design variables without artificial transformation into vector form and to develop a new hierarchical genetic algorithm for solving multilevel redundancy allocation optimization problems. The proposed methodology was tested on multilevel reliability problems of reasonable size and scope. In chapter 4, a modular concept in redundancy allocation was applied when solving reliability optimization problems in series and series-parallel systems. The modular redundancy scheme yields superior system reliability than traditional scheme of single level redundancy allocation with the same available resources. A second focus in this dissertation was to develop a methodology for multiobjective optimization of multilevel redundancy allocation problems. Multiobjective optimization is very useful when a designer faces a task to optimize several conflicting objectives and there are several conflicting objectives exist in optimal reliability design. A multiobjective formulation of multilevel redundancy allocation was proposed in chapter 5. A general framework of multiobjective hierarchical genetic algorithms was developed and applied to solve

multilevel redundancy allocation optimization problems.

6.1 Summary and conclusions

6.1.1 Hierarchy and modularity in optimal reliability design

In chapter 2, the concepts of hierarchy and modularity in system reliability design have been described and potential application of these well-proven techniques in reliability optimization is presented. The hierarchy helps to simplify the design of large scale systems and provide decomposability that helps to address the issue of managing the system more effectively throughout the life cycle. Since the structure of RBD affects the reliability optimization, the hierarchical concept of RBD is proposed in this chapter to simplify the design of complex system and represent exactly all the logical relationship between its subsystems and components. Similarly, the superiority of modular design is hard to challenge. The practical significance of the modular redundancy allocation in making a system more fault tolerant when so optimizing hierarchical RBD is explained.

Finally this chapter proposed a general formulation of multilevel redundancy allocation optimization problems. The proposed formulation has several novelties. This formulation allows redundancy allocation to all units at every level. Bi-level series and parallel modules is proposed as building blocks to represent all possible hierarchical RBD. Modular redundancy allocation can easily be applied when optimizing such hierarchical RBD.

6.1.2 Hierarchical genetic algorithms for MRAOP

In chapter 3, a general formulation for multilevel redundancy allocation optimization problems that aim to maximize system reliability is proposed. These multilevel optimization problems have hierarchical design variables, so we proposed a new coding method for use in a HGA, in which hierarchical design variables of MRAOP are represented using two types of hierarchical genotype: nodal, and terminal. We applied the

newly developed HGA, and a conventional GA separately, to solve two multilevel series redundancy allocation optimization problems having three, and four levels. The optimal solutions for these two problems demonstrated that the proposed HGA provides optimal system reliability that is superior to the conventional GA results, because it does not depend on the use of vector coding to represent the hierarchical variables, and can preserve the original design space.

6.1.3 Modular redundancy allocation optimization in series and series parallel system

In chapter 4, the importance of modular redundancy allocation applied to multilevel system reliability problems is discussed. We proposed a methodology to solve series and series-parallel redundancy allocation problems considering the hierarchical relationships among design variables. Modular design variables were encoded using hierarchical genotypes in hierarchical genetic algorithms, and the multilevel redundancy allocation optimization problems were efficiently solved. The optimization of numerical examples in this chapter indicates that the modular scheme of redundancy allocation yields superior system reliability for multilevel configurations, in contrast to the conventional notion that component level redundancy allocation yields better optimal solutions. The application of a HGA proved to be flexible and efficient when solving large-scale multilevel redundancy allocation optimization problems.

6.1.4 Multiobjective hierarchical genetic algorithms for MRAOP

In chapter 5, the multiobjective formulation of multilevel redundancy allocation problems and a general framework for multiobjective hierarchical genetic algorithms (MOHGAs) to solve them is proposed. Two multilevel redundancy allocation problems having 3 and 4 hierarchical levels were solved to maximize reliability while minimizing cost. The proposed MOHGAs with selection operators, NSGA-II and SPEA2, were applied to solve

the MRAOPs. For comparison with MO using MOHGA with NSGA-II and SPEA2 results, a SO optimization using a HGA was also applied to solve the MRAOPs. The results show that, for large multilevel problems, MO optimizations using MOHGA with NSGA-II are preferable to a SO optimization using a HGA because they provide superior solutions, as well as a Pareto set of optimal solutions that can be used during subsequent decision making. Additionally, the results show that the MOHGA with NSGA-II is superior to the MOHGA with SPEA2 for solving MRAOPs.

6.2 Recommendations for future works

6.2.1 Efficient optimization technique for solving MRAOPs

The numerical examples solved in chapter 3 and chapter 4 indicates that the size of search space in multilevel redundancy allocation optimization is very large and requires huge computational resources even in the case of the multilevel redundancy allocation problems with only four levels hierarchy. In practice, the hierarchy levels may go beyond four levels and search space size may be even larger. Therefore, we need to develop more efficient optimization techniques for solving multilevel redundancy allocation problems. This can be achieved by hybridization or parallelization of hierarchical genetic algorithms.

6.2.2 Selection operator in multiobjective hierarchical genetic algorithms

In chapter 5, the selection operator of multiobjective hierarchical genetic algorithm with NSGA-II provided superior solution than those with SPEA2. However, NSGA-II performance is not very encouraging in the case of multiobjective optimization with more than three objective functions. Also, practical problems involve more complex structures that are difficult to handle when optimizing multilevel redundancy allocation optimization. Building on the research carried out for this paper, the future research should focus on designing better selection strategies for multiobjective GAs which can effectively preserve

the structural diversity of large populations of individuals when searching for optimal solutions to large scale MRAOPs.

6.2.3 Reliability optimization of hierarchical network systems

Network problems arise frequently in communication systems and fault-tolerant designs are highly desirable for smooth data transfer. The hierarchical network optimization problem is the problem of finding the least cost network, with nodes divided into groups, edges connecting nodes in each groups and groups ordered in a hierarchy. The idea of hierarchical networks comes from telecommunication networks where hierarchies exist. Hierarchical networks can be designed and the corresponding mathematical models can be proposed. The problem is to maximize the reliability of whole network in such a way that which edges should connect nodes, and how demand is routed in the network. Such problem can be solved by decomposing hierarchical network problems into simple two-level substructure and applying the proposed hierarchical genetic algorithms.

6.2.4 Optimal design of *k*-out-of-*n* structure in a hierarchical system

Optimal design of a *k*-out-of-*n* system structure is an important issue to make fault-tolerant systems. This structure is defined as an *n*-component system that works if and only if at least *k*-out-of-*n* components work. It finds wide applications in both industrial and aerospace systems such as multi-display system in a cockpit, the multi-engine system in a airplane, and the multi-pump system in a hydraulic control system. However, for a large industrial system with thousand of subsystems and component, optimal design of *k*-out-of-*n* system is truly a technological challenge. The application of hierarchical and modular approach in optimal design of a *k*-out-of-*n* system for a large structure can simplify the design and ease the optimization process.

6.2.5 Optimal reliability design of time dependent hierarchical systems

This research used a constant reliability values for every module and component. However, this is impractical approach for the real world systems which are subject to several types of time-dependent stresses. Thus, every modules and component has different types of time-dependent failure patterns. In this dissertation, however, the life distributions of the components were not incorporated in the process of calculating the system reliability. In future research, time-dependency in the reliability function should be introduced. The models necessary to observe the reliability over the life of the system should be develop, instead of at just one point in time. In addition, performance measures, such as failure rate, MTTF and warranty time, should be estimated for the entire system. In other words, instead of dealing with R_i future research should use $R_i(t)$ in optimal reliability design of a large scale systems.

References

[1]. W. Kuo, V.R. Prasad, F.A. Tillman, C. Hwang. Optimal reliability design-fundamentals and applications. Cambridge University Press, 2001.

[2]. M. S. Chern. On computational complexity of reliability redundancy allocation in a series system. Opreation research letters, 11(5):309-315, 1992.

[3]. F. A. Tillman, C.L. Hwang, and W. Kuo. Optimization techniques for system reliability with redundancy – a review. IEEE Transactions on Reliability, R-26(26):148-155, 1977.

[4]. F. A. Tillman, C.L. Hwang, and W. Kuo. Optimization of system reliability. Marcel Dekker, New York, 1980.

[5]. K. B. Misra. On optimal reliability design: a review. System Science, 12(4): 5-30, 1986.

[6]. W. Kuo, V.R. Prasad. An annotated overview of system reliability optimization. IEEE Transactions on Reliability, R-49(2): 176-191, 2000.

[7]. K. K. Agrawal, K.B. Misra, and J.S.Gupta. A new heuristic criterion for solving a redundancy optimization problem. IEEE transaction on Reliability, R-24(24): 86-87, 1975.

[8]. S. K. Banarjee and K. Rajamani. Optimization of system reliability using a parametric approach. IEEE Transactions on Reliability, R-22(22): 35-39, 1973.

[9]. L. D. Bodin. Optimization procedure for the analysis of coherent structures. IEEE Transactions on Reliability, R-18(18): 118-126, 1969.

[10]. P.J. Boland, F. Proschan, and Y.L. Tong. Optimal arrangement of components via pair wise rearrangements. Naval Research Logistics Quarterly, 36(6): 807-814, 1989.

[11]. Y. Nakagawa, K. Nakashima, Y. Hittori. Optimal reliability allocation by branch and bound techniques. IEEE Transaction on Reliability, R-27(27): 31-38, 1978.

[12]. M. Sasaki. A simplified method of obtaining highest system reliability. In M.P. Smith editor, Proceeding of the Eighth National Symposium on Reliability and Quality Control, pp.489-502, Washington, D.C., 9-11 January, 1962.

[13]. P. M. Ghare and R.E. Taylor. Optimal redundancy for reliability in series system. Operation Research, 17(5):838-847, 1969.

[14]. Ir. R.N. von Hees and Ir. H. W. von den Meerendonk. Optimal reliability of parallel multi-coherent systems. Operation Research Quarterly, 12(1): 16-26, 1961.

[15]. G. E. Neuner and R.N. Miller. Resource allocation for maximum reliability. In L.S. Gephart, editor, Proceeding of 1966 Annual Symposium on Reliability, pp.332-346, San Francisco, CA, 25-27 January 1966.

[16]. K. K. Agrawal, K.B. Misra, and J.S. Gupta. Reliability evaluation-a comparative study of different techniques. Microelectronics and Reliability, 14(1):49-56, 1975.

[17]. R. Gordon. Optimum component redundancy for maximum system reliability. Operation Research, 5(2): 229-243, 1957.

[18]. K. Ida, M. Gen, and T. Yokota. System reliability optimization with several failure modes by genetic algorithm. 16th International Conference on Computers and Industrial Engineering, pp. 349–352, 1994.

[19]. S. G. Kneale. Reliability of parallel systems with repair and switching. In W.T. Sumerlin, editor, Proceeding of the Seventh National Symposium on Reliability and Quality Control, pp. 129-133, Philadelphia, PA, 9-11 January 1961.

[20]. F. A. Tillman. Optimization by integer programming of constrained reliability problems with several modes of failure. IEEE transactions on Reliability, R-18(18): 47-53, 1969.

[21]. J. M. Littschwager. Dynamic programming in the solution of a multistage reliability problem. Journal of Industrial Engineering , 15: 168-175, 1964.

[22]. L. A. Baxter and F. Harche. On the optimal assembly of series-parallel systems. Operations Research Letters, 11(3):153–157, 1992.

[23]. M. S. Chern, On the computational complexity of reliability redundancy allocation in a series system. Operations Research Letters, 11(5):309–315, 1992.

[24]. M. S. Chern and R. H. Jan. Parametric programming applied to reliability optimization problems," IEEE Transaction on Reliability, R-34(2):165–170, 1985.

[25]. M. S. Chern. Reliability optimization problems with multiple constraints. IEEE Transaction on Reliability, R-35(4): 431–436, 1986.

[26]. M. S. Chern, R. H. Jan, and R. J. Chern. Parametric nonlinear integer programming: The right-hand side case. European J. Operational Research, 54(2): 237–255, 1991.

[27]. D. W. Coit and A. Smith. Reliability optimization of series-parallel systems using a genetic algorithm. IEEE Transaction on Reliability, 45(2): 254–260, June 1996.

[28]. D. W. Coit and A. Smith. Solving the redundancy allocation problem using a combined neural network / genetic algorithm approach. Computers and Operations Research, 23(6): 515–526, June 1996.

[29]. D. W. Coit and A. Smith. Considering risk profiles in design optimization for series-parallel systems. In Proc. Annual Reliability and Maintainability Symposium, pp. 271–277, 1997.

[30]. A. K. Dhingra. Optimal apportionment of reliability & redundancy in series systems under multiple objectives. IEEE Transaction on Reliability, 41(4):576–582, 1992.

[31]. E. El-Neweihi, F. Proschan, and J. Sethuraman. Optimal allocation of components in parallel-series and series-parallel systems. Journal of Applied Probability, 23(3): 770–777, 1986.

[32]. E. El-Neweihi, F. Proschan, and J. Sethuraman. Optimal assembly of systems using Schur functions and majorization. Naval Research Logistics, 34: 705–712, 1987.

[33]. M. Gen, K. Ida, M. Sasaki, and J. U. Lee. Algorithm for solving large-scale 0-1 goal programming and its application to reliability optimization problem. Computers & Industrial Engineering, 17(1): 525–530, 1989.

[34]. K. Gopal, K. K. Aggarwal, and J. S. Gupta. An improved algorithm for reliability optimization. IEEE Transaction on Reliability, R-27(5): 325–328, 1978.

[35]. K. Gopal, K. K. Aggarwal, and J. S. Gupta. A new method for solving reliability optimization problem. IEEE Transaction on Reliability, R-29: 36–38, 1980.

[36]. K. K. Govil and R. A. Agarwala. Lagrange multiplier method for optimal reliability. Reliability Engineering, 6(3): 181–190, 1983.

[37]. F. K. Hwang and U. G. Rothblum. Optimality of monotone assemblies for coherent systems composed of series modules. Operations Research, 42(4): 709–720, 1994.

[38]. L. Jianping. A bound heuristic algorithm for solving reliability redundancy optimization. Microelectronics and Reliability, 3(5): 335–339, 1996.

[39]. W. Kuo. Reliability enhancement through optimal burn-in decision. IEEE Transaction on Reliability, R-33(2): 145–156, 1984.

[40]. W. Kuo, C. L. Hwang, and F. A. Tillman. A note on heuristic methods in optimal system reliability. IEEE Transaction on Reliability, R-27(5): 320–324, 1978.

[41]. W. Kuo, H. Lin, Z. Xu, and W. Zhang. Reliability optimization with the Lagrange multiplier and branch-and-bound technique. IEEE Transaction on Reliability, R-36(5): 624–630, 1987.

[42]. K. B. Misra. An algorithm to solve integer programming problems: An efficient tool for reliability design. Microelectronics and Reliability, 31(2/3): 285–294, 1991.

[43]. K. B. Misra and U. Sharma. An efficient algorithm to solve integer programming problems arising in system reliability design. IEEE Transaction on Reliability, 40(1): 81–91, 1991.

[44]. K. B. Misra and U. Sharma. An efficient approach for multiple criteria redundancy optimization problems. Microelectronics and Reliability, 31(2/3): 303–321, 1991.

[45]. K. B. Misra and U. Sharma. Multicriteria optimization for combined reliability and redundancy allocation in systems employing mixed redundancies. Microelectronics and Reliability, 31(2/3): 323–335, 1991.

[46]. Y. Nakagawa and S. Miyazaki. Surrogate constraints algorithm for reliability optimization problem with two constraints. IEEE Transaction on Reliability, R-30(2): 175–180, 1981.

[47]. Y. Nakagawa and S. Miyazaki. An experimental comparison of the heuristic methods for solving reliability optimization problems. IEEE Transaction on Reliability, R-30(2): 181–184, 1981.

[48]. Y. Nakagawa and K. Nakashima. A heuristic method for determining optimal reliability allocation. IEEE Transaction on Reliability, R-26(3): 156–161, 1977.

[49]. L. Painton and J. Campbell. Genetic algorithms in optimization of system reliability. IEEE Transaction on Reliability, vol. 44, pp. 172–178, 1995.

[50]. D. Petrovic. Decision support for improving systems reliability by redundancy. European Journal of Operational Research, 55(3): 357–367, 1991.

[51]. V. R. Prasad, Y. P. Aneja, and K. P. K. Nair. A heuristic approach to optimal assignment of components to a parallel-series network. IEEE Transaction on Reliability, 40(5): 555–558, 1991.

[52]. V. R. Prasad,W. Kuo, and K. M. O. Kim. Optimal allocation of s-identical multi-functional spares in a series system. IEEE Transaction on Reliability, 48(2): 118–126, 1999.

[53]. V. R. Prasad, K. P. K. Nair, and Y. P. Aneja. Optimal assignment of components to parallel-series and series-parallel systems. Operations Research, 39(3): 407–414,

1991.

[54]. V. R. Prasad and M. Raghavachari. Optimal allocation of interchangeable components to series-parallel reliability system. IEEE Trans. Reliability, 47(3): 255–260, 1998.

[55]. M. Sakawa. Optimal reliability-design of a series-parallel system by a large-scale multiobjective optimization method. IEEE Trans. Reliability, R-30(2): 173–174, 1981.

[56]. J. Sharma and K. V. Venkateswaran. A direct method for maximizing the system reliability. IEEE Transaction on Reliability, R-20(1): 256–259, 1971.

[57]. U. Sharma and K. B. Misra. An efficient algorithm to solve integer programming problems in reliability optimization. International Journal of Quality & Reliability Management, 7(5): 44–56, 1990.

[58]. F. A. Tillman. Optimization by integer programming of constrained reliability problems with several modes of failure. IEEE Transaction on Reliability, R-18(2): 47–53, 1969.

[59]. F. A. Tillman, C. L. Hwang, and W. Kuo. Determining component reliability and redundancy for optimum system reliability. IEEE Transaction on Reliability, R-26(3): 162–165, 1977.

[60]. F. A. Tillman, C. L. Hwang, and W. Kuo. Reliability optimization by generalized Lagrangian function and reduced gradient methods. IEEE Transaction on Reliability, R-28(4): 316–320, 1979.

[61]. Z. Xu, W. Kuo, and H. Lin. Optimization limits in improving system reliability. IEEE Transaction on Reliability, 39(1): 51–60, 1990.

[62]. T. Yokota, M. Gen, and K. Ida. System reliability of optimization problems with several failure modes by genetic algorithm. Japanese Journal of Fuzzy Theory and

Systems, 7(1): 117–135, 1995.

[63]. T. Yokota, M. Gen, and Y. X. Li. Genetic algorithm for nonlinear mixed integer programming problems and its applications. Computers & Industrial Engineering, 30(4): 905–917, 1996.

[64]. D. H. Chi and W. Kuo. Burn-in optimization subject to reliability and capacity constraints. IEEE Transaction on Reliability, 38(3): 193–198, 1989.

[65]. D. L. Deeter and A. E. Smith. Economic design of reliable network. IIE Transactions, 30(12): 1161–1174, 1998.

[66]. S. Dinghua. A new heuristic algorithm for constrained redundancy-optimization in complex systems. IEEE Transaction on Reliability, R-36(5): 621–623, 1987.

[67]. M. Hikita, Y. Nakagawa, K. Nakashima, and H. Narihisa. Reliability optimization of systems by a surrogate-constraints algorithm. IEEE Transaction on Reliability, 41(3): 473–480, 1992.

[68]. J. H. Kim and B. J. Yum. A heuristic method for solving redundancy optimization problems in complex systems. IEEE Transaction on Reliability, 42(4): 572–578, 1993.

[69]. J. Y. Kim and L. C. Frair. Optimal reliability design for complex systems IEEE Transaction on Reliability, R-30(2): 300–302, 1981.

[70]. T. Kohda and K. Inoue. A reliability optimization method for complex systems with the criterion of local optimality. IEEE Transaction on Reliability, R-31(1): 109–111, 1982.

[71]. W. Kuo, C. L. Hwang, and F. A. Tillman. A note on heuristic methods in optimal system reliability. IEEE Transaction on Reliability, R-27(5): 320–324, 1978.

[72]. D. Li. Interactive parametric dynamic programming and its application in reliability optimization. Journal of Mathematical Analysis and Applications, 191(5): 589–607,

1995.

[73]. D. Li and Y. Y. Haimes. A decomposition method for optimization of large system reliability. IEEE Transaction on Reliability, 41(2): 183–188, 1992.

[74]. C. Mohan and K. Shanker. Reliability optimization of complex systems using random search technique. Microelectronics and Reliability, 28(4): 513–518, 1988.

[75]. C. A. Ntuen, E. H. Park, and W. Byrd. A heuristic program for reliability and maintainability allocation in complex hierarchical systems. Computers and Industrial Engineering, 25(1-4): 345–348, 1993.

[76]. V. Ravi, B. Murty, and P. Reddy. Nonequilibrium simulated-annealing algorithm applied reliability optimization of complex systems. IEEE Transaction on Reliability, 46(2): 233–239, 1997.

[77]. U. Sharma, K. B. Misra, and A. K. Bhattacharjee. Application of an efficient search technique for optimal design of computer communication network. Microelectronics and Reliability, 31(2/3): 337–341, 1991.

[78]. D. S. Bai, W. Y. Yun, and S. W. Cheng. Redundancy optimization of k-out-of-n:G systems with common-cause failures. IEEE Transaction on Reliability, 40(2): 56–59, 1991.

[79]. C. Derman, G. J. Lieberman, and S. M. Ross. On the consecutive k-out-of-n:F system. IEEE Transaction on Reliability, R-31(1): 57–63, 1982.

[80]. D. Z. Du and F. K. Hwang. Optimal consecutive 2-out-of-n systems. Mathematics of Operations Research, 11(1) 187–191, 1986.

[81]. D. Z. Du and F. K. Hwang. Optimal assembly of an s-stage k-out-of-n system. SIAM Journal of Discrete Mathematics, 3(3): 349–354, 1990.

[82]. E. El-Neweihi, F. Proschan, and J. Sethuraman. Optimal assembly of systems using Schur functions and majorization. Naval Research Logistics, 34:705–712, 1987.

[83]. F. K. Hwang. Optimal assignment of components to a two-stage k-out-of-n system. Mathematics of Operations Research, 14(2): 376–382, 1989.

[84]. F. K. Hwang and S. Dinghua. Redundant consecutive-k systems. Operations Research Letters, 6(6): 293–296, 1987.

[85]. D. M. Malon. Optimal consecutive 2-out-of-n: F component sequencing. IEEE Transaction on Reliability, R-33(5): 414–418, 1984.

[86]. V. K. Wei, F. K. Hwang, and V. T. Sos. Optimal sequencing of items in a consecutive 2-out-of-n system. IEEE Transaction on Reliability, R-32(1): 30–34, 1983.

[87]. M. J. Zuo andW. Kuo. Design and performance analysis of consecutive k-out-of-n structure. Naval Research Logistics Quarterly, 37(2): 203–230, 1990.

[88]. F. Belli and P. Jedrzejowicz. An approach to the reliability optimization of software with redundancy. IEEE Transaction on Software Engineering, 17(3): 310–312, 1991.

[89]. O. Berman and N. Ashrafi. Optimization models for reliability of modular software systems IEEE Transaction on Software Engineering, 19(11): 1119–1123, 1993.

[90]. D. H. Chi and W. Kuo. Optimal design for software reliability and development cost. IEEE Journal of Selected Areas in Communications, 8(2): 276–281, 1990.

[91]. C. J. Dale and A. Winterbottom. Optimal allocation of effort to improve system reliability. IEEE Transaction on Reliability, R-35(2): 188–191, 1986.

[92]. H. Pham. On the optimal design of n-version software systems subject to constraints. Journal of Systems & Software, 27(1): 55–61, 1994.

[93]. V. R. Prasad and W.Kuo. Reliability optimization of coherent systems. IEEE Trans. Reliability, R-49(3): 2000.

[94]. D.H. Shi. A new heuristic algorithm for constrained redundancy-optimization in complex systems. IEEE Transactions on Reliability, R-36(36): 621-623, 1987.

[95]. K.B. Misra. A simple approach for constrained redundancy optimization problems.

IEEE Transactions on Reliability, R-21(21): 30-34, 1972.

[96]. K. Misra and V. Misra. A procedure for solving general integer programming problems. Microelectronics and Reliability, 34(1): 157-163, 1994.

[97]. M. Sakawa. Interactive multiobjective optimization by sequential proxy optimization technique. IEEE Transactions on Reliability, R-31(31):461-464, 1982.

[98]. R.E. Bellman and S.E. Dreyfus. Dynamic Programming and Reliability of Multi-component devices. Operation Research, 6:200-206, 1958.

[99]. R.M. Burton and G.T. Howard. Optimal system reliability for a mixed series and parallel structure. Journal of Mathematical Analysis and Applications, 28(2): 370-382, 1969.

[100]. D.K. Kulshreshtha and M.C. Gupta. Use of dynamic programming for reliability engineers. IEEE Transactions on Reliability, R-22(22): 240-241, 1973.

[101]. B.K. Lambert, A.G. Walvekar, and J.P. Hirmas. Optimal redundancy and availability allocation in multistage systems. IEEE Transactions on Reliability, R-20(20): 182-185, 1971.

[102]. M. Gen, H. Okuno, and S. Shinofuji. An optimizing method in system reliability with failure modes by implicit enumeration algorithm. Journal of Operation Research, Japan, 19(2): 99-116, 1976.

[103]. C. L. Hwang, L. T. Fan, F. A. Tillman, and S. Kumar. Optimization of life support system reliability by an integer programming method. AIIE Transactions, 3(3): 229-238, 1971.

[104]. K. N. Hyun. Reliability optimization by 0-1 programming for a system with several failure modes. IEEE Transactions on Reliability, R-24(24): 206-210, 1975.

[105]. P. J. Kolesar. Linear programming and the reliability of multi-component systems. Naval Research Logistics Quarterly, 14(3):317-327, 1967.

[106]. E.L. Lawler and M.D. Bell. A method for solving discrete programming problems. Operations Research, 14 (6):1089–1112, 1966.

[107]. R. Luus. Optimization of system reliability by a new nonlinear integer programming procedure. IEEE Transactions on Reliability, R-24(24): 14-16, 1975.

[108]. U. Sharma and K. B. Misra. An efficient algorithm to solve integer-programming problems in reliability optimization. International Journal of Quality and Reliability Management 7(5): 44-56, 1996.

[109]. Filus Jerzy. A problem in reliability optimization. Journal of the operation research society. 37(4): 407-412, 1986.

[110]. Y. Nakagawa and S. Miyazaki. Surrogate constraints algorithm for reliability optimization problem with two constraints. IEEE Transactions on Reliability, R-30(30): 175-180, 1981.

[111]. K. K. Govil and R. A. Agarwala. Langrange multiplier method for optimal reliability allocation in a series system. Reliability Engineering, 6(3): 181-190, 1983.

[112]. M. S. Chern, and R. H. Jan. Parametric programming applied to reliability optimization problems. IEEE Transactions on Reliability, R-34(34): 165-170, 1985.

[113]. M. S. Chern, and R. H. Jan. Parametric nonlinear integer programming: the right-hand-side case. European Journal of Operation Research, 54(2): 237-255, 1991.

[114]. L. A. Baxter and F. Harche. On optimal assembly of series-parallel systems. Operation Research Letters, 11(3): 153-157, 1992.

[115]. D. Chi and Way Kuo. Burn-in optimization under reliability and capacity restrictions. IEEE Transactions on Reliability, R-38(2): 193-198, 1989.

[116]. C. L. Hwang, K. C. Lai, F. A. Tillman, and L. T. Fan. Optimization of system

reliability using sequential unconstrained minimization technique. IEEE Transaction on Reliability, R-24(24):133-135, 1975.

[117]. H. V. K. Shetty and D. P. Sengupta. Reliability optimization using SUMT. IEEE Transaction on Reliability, R-24(24):80-82, 1975.

[118]. H. Everett III. Generalized Langrange multiplier method for solving problems of optimal allocation of resources. Operation Research , 11(3): 399-417, 1963.

[119]. K.B. Misra. Reliability Optimization of a series-parallel system. PartI: Langrange multiplier approach; Part II: maximum principle approach. IEEE Transaction on Reliability, R-21(21):230-238, 1972.

[120]. D. S. Necsulescu and M. Krieger. Reliability optimization-a case study. IEEE Transactions on Reliability, R-31(31): 101-104, 1982.

[121]. G. H. Holland. Adaption in natural and artificial systems. University of Michigan Press, 1975.

[122]. D.E. Goldberg. Genetic Algorithms in search, optimization & Machine Learning. Addition-Wisley, New York, 1989.

[123]. S. Kirkpatrick, C.D. Gelat, and M.P. Vecchi. Optimization by Simulated Annealing. Science, 220(4598): 671-680, 1983.

[124]. F. Glover. Future Paths for Integer Programming and Links to Artificial Intelligence. Computers and Operation Research, 13(5): 533-549, 1986.

[125]. J. Kennedy and R.C. Eberhart. Particle Swarm Optimization. Proceedings of the 1995 IEEE International Conference on Neural Networks, 4:1942-1948, 1995.

[126]. M. F. Cardoso, R. L. Salcedo, and S.F. de Azevedo. Nonequilibrium simulated annealing: a faster approach to combinatorial minimization. Industrial and Engineering Chemistry Research, 33(8): 1908-1918, 1994.

[127]. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. Teller, and E. Teller.

Equation of state calculations by fast computing machines. Journal of Chemical Physics, 21: 1087-1092, 1953.

[128]. M. Gen and R. Cheng, Genetic Algorithms and Engineering Design. Wiley, New York, 1997.

[129]. M. Gen, and Y. S. Yun. Soft computing approach for reliability optimization: state-of-the-art survey. Reliability Engineering & System Safety, 91(9): 1008-1026, 2006.

[130]. M.Gen, J.R. Kim. Bicriteria reliability design using hybrid genetic algorithm. In: Proceedings of the sixth European congress on intelligent techniques and soft computing; pp.407-12, 1998.

[131]. C.Y. Lee, M. Gen and W. Kuo. Reliability optimization design using a hybridized genetic algorithm with a neural-network technique. IEICE Transaction on Fundamentals, E84-A (2):627–635, 2001.

[132]. C.Y. Lee, M. Gen and Y. Tsujimura. Reliability optimization design using hybrid NN-GA with fuzzy logic controller. IEICE Transaction on Fund E85-A (2):432–446, 2002.

[133]. Y.S. Yun, M. Gen and S.L. Seo. Various hybrid methods based on genetic algorithm with fuzzy logic controller. Journal of Intelligent Manufacturing, 14 (3–4): 401–419, 2003.

[134]. C.Y. Lee, Y.S. Yun and M. Gen. Reliability optimization design for complex systems by hybrid GA with fuzzy logic control and local search. IEICE Transaction of Fundamentals, E85-A (4):880–891, 2002.

[135]. H. Simon. The Sciences of the Artificial. MIT press: Cambridge, 1968.

[136]. M. Pelika and D.E. Goldberg. A Hierarchy Machine: Learning to optimize from Nature and Humans. Complexity, 8(5):36-45, 2003.

[137]. N. Rasmussen and S. Niles. Modular systems: the evolution of reliability. White paper, 76. American Power Corporation, 2005.

[138]. W. Wang, N.J. Loman, and P. Vassiliou. Reliability importance of components in a complex system. Proceedings of the annual Reliability and Maintainability Symposium, pp.6-11, LA, 2004.

[139]. I. Koren and Z.Koren. Defect tolerance in VLSI circuits: techniques and yield analysis. Proceedings of the IEEE, 86(9):1819-1837, 1998.

[140]. G. Levitin. Optimal multilevel protection in series–parallel systems. Reliability Engineering & System Safety, 81 (1):93–102, 2003.

[141]. S. H. Baek, B. W. Kim, E. J. Joung, and C. W. Park. Reliability and performance of hierarchical RAID with multiple controllers. Proceedings of the twentieth annual ACM symposium on Principles of distributed computing, pp.246 – 254, Newport, Rhode Island, 2001

[142]. C. Ha and W. Kuo. Reliability redundancy allocation: An improved realization for nonconvex nonlinear programming problems. European Journal of Operational Research, 171(1):24-38, 2006.

[143]. W. Kuo and M.J. Zuo. Optimal Reliability Modeling: Principles and Applications. Wiley, New York, 2003.

[144]. P. Boland and E. EL-Neweihi. Component redundancy vs. system redundancy in the hazard rate ordering. IEEE Transactions on Reliability, 44(4): 614–619, 1995.

[145]. G. Anandlingam and T.L. Friesz. Hierarchical Optimization: An Introduction. Annals of Operations Research, 34(1-4):1-11, 1992

[146]. L. Painton and J. Campbell. Genetic Algorithms in optimization of system reliability. IEEE Transaction on Reliability, R-44 (2):172-178, 1995.

[147]. T. Yokata, M. Gen, and K. Ida. System reliability of optimization problems with

several failure modes by genetic algorithm. Japanese Journal of Fuzzy Theory and Systems, 7 (1):117-135, 1995.

[148]. K. Ida, M. Gen, and T. Yokata. System reliability optimization with several failure modes by genetic algorithm. Proceedings of the 16th International Conference on Computers and Industrial Engineering, pp. 349-352, Ashikaga, Japan, 1994.

[149]. D.W. Coit and A.E. Smith. Reliability optimization of series-parallel systems using a genetic algorithm. IEEE Transactions on Reliability, R-45 (2):254-260, 1996.

[150]. D.W. Coit and A.E. Smith. Considering risk profiles in design optimization for series-parallel systems. Proceedings of the Annual Reliability and Maintainability Symposium, pp.271-277, Philadelphia, PA, 1997.

[151]. Y.C. Hsieh, T.C. Chen, and D.L. Bricker. Genetic algorithms for reliability design problems. Technical Report, Department of Industrial Engineering, University of Iowa, 1997.

[152]. W.Y. Yun and J.W. Kim. Multi-Level redundancy optimization in series systems. Computers & Industrial Engineering, 46(2): 337–346. 2004

[153]. E.D. DeJong, D. Thierens, and R.A. Watson. Hierarchical Genetic Algorithms. Parallel Problem Solving From Nature-PPSNVIII, pp. 232-241, 2004.

[154]. J. Hu and E. D. Goodman. The hierarchical fair competition (HFC) model for parallel evolutionary algorithms. Proceedings of the Congress on Evolutionary Computation pp 49–54, 2002.

[155]. M. Gulsen, and A. E. Smith. A hierarchical genetic algorithm for system identification and curve fitting with a supercomputer implementation. Evolutionary Algorithms. Springer, pp.111–137, New York, 1999

[156]. K. Tang, K. Man and R. Istepanian. Teleoperation controller design using hierarchal genetic algorithms. Proceedings of the IEEE International conference on Industrial

Technology, pp.707–711, 2000.

[157]. M. Sefrioui and J. Périaux. A hierarchical genetic algorithm using multiple models for optimization. Proceedings of the 6th International Conference on Parallel Problem Solving from Nature, pp.879 – 888, 2000.

[158]. M. Yoshimura, K. Izui. Smart optimization of machine systems using hierarchical genotype representations. ASME Journal of Mechanical Design, 124(3): 375-384, 2002.

[159]. M. Gen and R. Cheng. A survey of penalty technique in genetic algorithms. Proceedings of the International Conference on Evolutionary Computation, pp.804-809, Nagoya University, Japan, 1996.

[160]. A. Molina, A. Kusiak, and J. Sanchez. Handbook of life cycle engineering: concepts, models and technologies. Kluwer Academic Publishers, Boston, 1999.

[161]. W. Y. Yun, Y. M. Song, and H. G. Kim. Multiple multi-level redundancy allocation in series systems. Reliability Engineering and Systems Safety, 92(3): 308-313, 2007.

[162]. R. Kumar, K. Izui, M. Yoshimura, and S. Nishiwaki. Multilevel redundancy allocation optimization using a hierarchical genetic algorithm. IEEE Transaction on Reliability (in press), 2008.

[163]. K. Deb. Multiobjective optimization using evolutionary algorithms. Chichester, UK: John wiley and sons; 2001.

[164]. K., Deb, A. Pratap, S. Agrawal, T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transaction on Evolutionary Computation, 6(2):182-197, 2002.

[165]. E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm. TIK Report No. 103, Swiss Federal Institute of Technology;

2001.

[166]. E. Zizler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Transaction on Evolutionary Computation, 3(4): 257-271, 1999.

[167]. E. D. DeJong, D. Thierens, and R. A. Watson. Hierarchical genetic algorithms. In: Proceedings of eighth international conference on parallel problem solving from Nature, September 18–22, Burmingham, UK: Springer; 2004.

[168]. A. Konak, D. W. Coit, and A. E. Smith. Multi-objective optimization using genetic algorithms: A tutorial. Reliability Engineering and System Safety, 91(9):992-10072006.

[169]. E Zizler, K Deb, and L Thiele. Comparision of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation, 8(1):173-195, 2000.

[170]. J.D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the first international conference on genetic algorithm and their applications. July 24-26, 1985. Pittsburgh, Pa, USA: Lawrence Erlbaum Associates; 1985.

[171]. J. Horn, N. Nafpliotis, D. E. Goldberg. A niched Pareto genetic algorithm for multiobjective optimization. In: Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence. June 27–29, 1994. Orlando, FL, USA: IEEE; 1994.

[172]. P. Hajela, C. Y. Lin. Genetic search strategies in multicriterion optimal design. Structural Optimization, 4(2):99-107, 1992.

[173]. T. Murata and H. Ishibuchi. MOGA: multi-objective genetic algorithms. In: Proceedings of the IEEE international conference on evolutionary computation, November 29- December 1, 1995. Perth, WA, Australia: IEEE; 1995.

[174]. N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. Journal of Evolutionary Computation, 2(3):221–248, 1994.

[175]. C. M. Fonseca and P. J. Fleming. Multiobjective genetic algorithms. In: IEEE colloquium on 'Genetic Algorithms for Control Systems Engineering' May 28, 1993. London, UK: IEE; 1993.

[176]. J. D. Knowles and D. W. Corne. Approximating the nondominated front using the Pareto archived evolution strategy. Evolutionary Computation, 8(2):149–172, 2000..

[177]. D. W. Corne, J. D. Knowles, and M. J. Oates. The Pareto envelope-based selection algorithm for multiobjective optimization. In: Proceedings of sixth international conference on parallel problem solving from Nature, September 18–20, 2000. Paris, France: Springer; 2000.

[178]. D. Corne, N. R. Jerram, J. Knowles, J. Oates. PESA-II: region-based selection in evolutionary multiobjective optimization. In: Proceedings of the genetic and evolutionary computation conference, July 7-11, 2001. San Francisco, CA, USA: 2001.

[179]. R. Sarker, K. H. Liang, and C. Newton. A new multiobjective evolutionary algorithm. European Journal of Operation Research, 140(1):12-23, 2002.

[180]. C. A. C. Coello and G. T. Pulido. A micro-genetic algorithm for multiobjective optimization. In: First international conference on 'Evolutionary multi-criterion optimization (EMO 2001). March 7-9, 2001. Zurich, Switzerland: Springer; 2001.

[181]. H. Lu and G. G. Yen. Rank-density-based multiobjective genetic algorithm and benchmark test function study. IEEE Transaction on Evolutionary Computation, 7(4):325-343, 2003.

[182]. G. G. Yen and H. Lu. Dynamic multiobjective evolutionary algorithm: adaptive cell-based rank and density estimation. IEEE Transaction on Evolutionary

Computation, 7(3):253-274, 2003.

[183]. M. Yoshimura, M. Taniguchi, K. Izui, and S. Nishiwaki. Hierarchical arrangement of characteristics in product design optimization. ASME Journal of Mechanical Design, 128 (4):701-709, 2006.

Acknowledgement

First, I would like to express my sincere gratitude to my supervisor, Professor Masataka Yoshimura, for his support, encouragement, and guidance during my stay at Kyoto University. He gave a lot of freedom in my course and research work, and has been extremely supportive and understanding at all times. Second, I would especially like to thank Professor Kazuhiro Izui, who has supported all of my research activities. He has been a constant source of help and inspiration to me. His guidance shaped my research activities and helped me to complete this research and write the dissertation. Also, I am extending my thanks and appreciation to Professor Shinji Nishiwaki for providing all the crucial guidance and supports since the commencement of my research.

I appreciate the administrative help provided by the laboratory administrative assistant Ms. Hiromi Ishizuka. I would like to extend my thanks to my tutor Mr. Haruki Karia for helping me settle in Kyoto University and learn basic language skill. I would like to thank all the lab members, Mr. Kiyoshi Yokota, Mr. Shin Kikuchi, Mr. Naotaka Uchida, Mr. Keiichi Noda of Yoshimura laboratory for providing highly conducive research environment in the laboratory. Also, I would like to thank Mr. Kenji Doi for his insightful discussion and constant encouragement to complete my research.

Last but not the least, I thank my parent and the other members of my family whose support and understanding is a continuous source of encouragement. I thank housemother, Mrs. Maekawa Kayoko, office staffs, and all co-residents of the Kyoto International Student House for providing a homely environment throughout my staying period.

Related Works

[1] R. Kumar, K. Izui, M. Yoshimura, and S. Nishiwaki. Multilevel redundancy allocation optimization using a hierarchical genetic algorithm. IEEE Transaction on Reliability, 57(4):650-661, 2008.

[2] R. Kumar, K. Izui, M. Yoshimura, and S. Nishiwaki. Optimal multilevel redundancy allocation in series and series-parallel systems. Computers and Industrial Engineering, (in press).

[3] R. Kumar, K. Izui, M. Yoshimura, and S. Nishiwaki. Multiobjective hierarchical genetic algorithms for multilevel redundancy allocation optimization. Reliability Engineering and System Safety, (in press).

[4] R. Kumar, K. Izui, M. Yoshimura, and S. Nishiwaki. Hierarchical Genetic Algorithm for System Reliability Optimization. The 7th International Conference on Optimization: Techniques and Applications (ICOTA7) December 12 - 15, 2007, Kobe International Conference Center, Kobe, Japan.

[5] R. Kumar, K. Izui, M. Yoshimura, and S. Nishiwaki. Optimal Analysis of Hierarchical Redundancy Allocation Using Competent Genetic Algorithms. 7th World Congress on Structural and Multidisciplinary Optimization, 2007, Seoul, South Korea.

[6] R. Kumar, K. Izui, M. Yoshimura, and S. Nishiwaki. Modular Redundancy Optimization Using Hierarchical Genetic Algorithms. Annual Reliability and Maintainability Symposium, 2007 Orlando, Florida USA.

[7] R. Kumar, K. Izui, M. Yoshimura, and S. Nishiwaki. Robustness of Redundancy Optimization using Hierarchical Genetic Algorithms. The Japanese Society of Mechanical Engineering, D&S 2006 Symposium, Nagoya, Japan.

[8] R. Kumar, K. Izui, M. Yoshimura, and S. Nishiwaki. Hierarchical Reliability Optimization using Competent Genetic Algorithms. The Fourth China-Japan-Korea

Joint Symposium on Optimization of Structural and Mechanical Systems Nov. 6-9, 2006, Kunming, China.